# Concurrent Multipath Transfer: Scheduling, Modelling, and Congestion Window Management

(Spine title: CMT: Scheduling, Modelling, and CWND Management)

(Thesis format: Monograph)

by

Thomas Daniel Wallace

Graduate Program
in
Engineering
Electrical and Computer Engineering

A thesis submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

# Certificate of Examination

THE UNIVERSITY OF WESTERN ONTARIO
SCHOOL OF GRADUATE AND POSTDOCTORAL STUDIES
**CERTIFICATE OF EXAMINATION**

**Chief Advisor:**

Dr. Abdallah Shami

**Advisory Committee:**

**Examining Board:**

Dr. Miriam Capretz

Dr. Raveendra Rao

Dr. Hanan Lutfiyya

Dr. Nasir Ghani

The thesis by
**Thomas Daniel Wallace**
entitled:
**Concurrent Multipath Transfer: Scheduling, Modelling, and Congestion Window Management**
is accepted in partial fulfillment of the
requirements for the degree of
**Doctor of Philosophy**

Date: _____

Chair of Examining Board

# Abstract

Known as *smartphones*, multihomed devices like the iPhone and BlackBerry can simultaneously connect to Wi-Fi and 4G LTE networks. Unfortunately, due to the architectural constraints of standard transport layer protocols like the transmission control protocol (TCP), an Internet application (e.g., a file transfer) can use only one access network at a time. Due to recent developments, however, concurrent multipath transfer (CMT) using the stream control transmission protocol (SCTP) can enable multihomed devices to exploit additional network resources for transport layer communications.

In this thesis we explore a variety of techniques aimed at CMT and multihomed devices, such as: packet scheduling, transport layer modelling, and resource management. Some of our accomplishments include, but are not limited to: enhanced performance of CMT under delay-based disparity, a tractable framework for modelling the throughput of CMT, a comparison of modelling techniques for SCTP, a new congestion window update policy for CMT, and efficient use of system resources through optimization.

Since the demand for a better communications system is always on the horizon, it is our goal to further the research and inspire others to embrace CMT as a viable network architecture; in hopes that someday CMT will become a standard part of smartphone technology.

**Keywords:** multihoming, stream control transmission protocol, concurrent multipath transfer, transport layer, smartphones, mathematical modelling

# Acknowledgements

Throughout this journey there have been two people that have supported me the most; without them, this thesis would not have been possible.

First, I'd like to thank my supervisor and friend Dr. Abdallah Shami, not only for giving me the opportunity to become a graduate student at the University of Western Ontario, but also seeing me through to the end of my PhD. Every time (and there were many) I doubted myself, I only needed to speak with Dr. Shami briefly before reestablishing confidence and getting on with what needed to be done. No matter what the circumstances, I know I can always rely on him to boost my spirits and set me on a course for success.

The second person that needs to be recognized along with this achievement is my "life partner", Jamie. I can only imagine the mess I would be in if Jamie were not in my life. Jamie really is the best thing to have ever happened to me and she deserves all the credit.

# Table of Contents

# List of Tables

# List of Figures

# Acronyms

| | |
|---|---|
| **ACK** | *acknowledgement* |
| **BAS** | *bandwidth aware scheduler* |
| **BDP** | *bandwidth delay product* |
| **CA** | *congestion avoidance* |
| **CDMA** | *code division multiple access* |
| **CMT** | *concurrent multipath transfer* |
| **CUMACK** | *cumulative acknowledgement* |
| **CWND** | *congestion window* |
| **DAR** | *dynamic address reconfiguration* |
| **EB** | *exponential back-off* |
| **ECN** | *explicit error notification* |
| **FIFO** | *first in first out* |
| **FR** | *fast recovery* |
| **GSM** | *Global System for Mobile Communications* |
| **IETF** | *Internet Engineering Task Force* |
| **ILP** | *integer linear programming* |
| **IP** | *Internet protocol* |
| **IS-IS** | *intermediate system to intermediate system* |
| **ISP** | *Internet service provider* |
| **LAN** | *local area network* |
| **LTE** | *long term evolution* |
| **MAC** | *media access control* |
| **MAN** | *metropolitan area network* |
| **MTU** | *maximum transmission unit* |
| **ODS** | *on-demand scheduler* |
| **OSPF** | *open shortest path first* |

| | |
|---|---|
| **PBA** | *partial.bytes.acked* |
| **PMR** | *path.max.retrans* |
| **QoS** | *quality of service* |
| **RBUF** | *receive buffer* |
| **RSS** | *received signal strength* |
| **RTO** | *retransmission timeout* |
| **RTT** | *round trip time* |
| **RWND** | *receive window* |
| **SACK** | *selective acknowledgement* |
| **SBUF** | *send buffer* |
| **SCTP** | *stream control transmission protocol* |
| **SS** | *slow-start* |
| **SSN** | *stream sequence number* |
| **SSTHRSH** | *slow-start threshold* |
| **TCP** | *transmission control protocol* |
| **TO** | *timeout* |
| **TSN** | *transmission sequence number* |
| **UDP** | *user datagram protocol* |
| **UMTS** | *Universal Mobile Telecommunications System* |
| **WiMAX** | *Worldwide Interoperability for Microwave Access* |
| **WLAN** | *wireless local area network* |

# Chapter 1
# Introduction

In just over a decade, mobile telephones have evolved from simple machines, offering no more than just voice communication, to highly advanced multimedia devices providing users with a rich set of services like: email, instant messaging, web applications, gaming, and even video conferencing. These new devices, known as *smartphones*, have changed the way we live our lives. No longer are we tethered to a desktop computer, or burdened from carrying a laptop; the smartphone has miniaturized the personal computer so we can shop, pay bills, listen to music, game, watch movies, or even work, at anytime and in anyplace from a machine small enough to fit into a pocket.

Unlike the desktop computer, a smartphone's resources are scarce. For example, a smartphone's storage capacity is extremely limited, so larger files are normally distributed over the Internet and downloaded only when needed. Networking and communications are therefore imperative to the evolution and sustainability of smartphone usage. At present, most smartphones connect to the Internet through a wireless local area network (WLAN) or cellular network. In either case, the underlying technology for communications (e.g., radio, media access control) is different; leading to separate network interfaces built into a single device[1].

Adding more than one network interface to a smartphone, or any computing device for that matter, is called multihoming. Devices that are multihomed are

1. The BlackBerry and iPhone are two popular smartphones; standard features include 802.11 technology for WLAN, and either GSM or UMTS technologies for cellular connectivity.

advantageous as they can provide a layer of redundancy in case of network failure (e.g., service disruption). Furthermore, one network may outperform another depending on location. For instance, when a smartphone user is at home, he may choose to connect to the Internet through a WLAN for higher data rates but lower fees. Alternatively, the cellular network is best while users are in transit. The larger coverage area of a cellular network can offer guarantees on connectivity, but at a much higher price with lower bandwidth potential.

Assuming either network's (i.e, WLAN or cellular) data rates are independent, so that one does not affect the other; why then do we not use both network interfaces simultaneously? Certainly, if two men could do the work of one in half the time, the same could be said about interfaces and download times. Unfortunately, due to the architectural constraints of standard transport layer protocols like the transmission control protocol (TCP), an Internet application (e.g., file transfer) can only use one access network at a time.

Due to recent developments, however, concurrent multipath transfer (CMT) using the stream control transmission protocol (SCTP) can exploit multihomed devices to enhance data communications. While SCTP is a new transport layer protocol that supports end-points with multiple IP addresses (i.e., a multihomed device), CMT provides a framework so that transport layer resources are used efficiently and effectively when sending to the same destination with multiple IP addresses. In this thesis, we study the following techniques for CMT: packet scheduling, transport layer modelling, and resource management. The goal is a better understanding of transport layer multihoming as well as improved performance of CMT.

## 1.1 Outline

This thesis consists of seven chapters. The remaining six chapters are organized as follows: Chapter 2 provides a comprehensive review of transport layer multihoming using SCTP; Chapter 3 introduces the system of study that is used extensively throughout the thesis; Chapter 4 investigates scheduling algorithms used for CMT and proposes the on-demand scheduler (ODS); Chapter 5 presents two mathematical models to approximate the average throughput of CMT; Chapter 6 introduces the concept of congestion window management and uses the models from Chapter 5 to solve an optimization problem to improve the performance of CMT; finally, Chapter 7 concludes this thesis and suggests future directions for extending the research.

## 1.2 Contributions

All contributions of the thesis are listed below.

### 1.2.1 Contributions of Chapter 2

**(1)** A comprehensive survey presenting the current research of SCTP as it pertains to multihomed devices is thoroughly discussed.

**(2)** Three main research areas for multihoming using SCTP are identified: handover management, concurrent multipath transfer, and cross-layer activities.

### 1.2.2 Contributions of Chapter 3

**(1)** A generalized network topology for a multihomed system is presented.

**(2)** A transport layer architecture for multihomed end-points is proposed. The new architecture includes a scheduler and destination manager.

### 1.2.3 Contributions of Chapter 4

**(1)** New insight into the effects of congestion and flow control on the scheduling process for CMT are revealed.

**(2)** A new scheduler for CMT, called the on-demand scheduler (ODS), is proposed. ODS differs from its predecessors in that it waits until congestion and flow control allow transmission before assigning a packet to a destination address.

### 1.2.4 Contributions of Chapter 5

**(1)** A tractable framework for modelling CMT is established by assuming the transport layer resources of each destination address are independent. Moreover, perfect scheduling is assumed so that the effects of reordering at the receiver can be ignored.

**(2)** Two mathematical models are presented that approximate throughput: the first uses a Markov chain and solves for a steady-state probability distribution; the second model employs renewal theory as a means to estimate the average outcome of a repeating stochastic process.

### 1.2.5 Contributions of Chapter 6

**(1)** A new concept called congestion window management is adopted as a main component of CMT. The congestion window manager has two responsibilities: limit

a destination's congestion window to the bandwidth potential of its corresponding network path, and configure the size of each congestion window to maximize aggregated throughput.

**(2)** A new congestion window update policy is developed to use transport layer resources more efficiently.

**(3)** Static and dynamic methods are proposed for congestion window optimization. While the dynamic method uses greedy optimization, the static approach employs an integer linear program (ILP) to solve the problem from a global standpoint.

**(4)** A heuristic was created to generate feasible solutions for the static congestion window management problem in a shorter amount of time than solving for optimality.

# Chapter 2

# A Review of Multihoming Issues using SCTP [1]

In recent years, advancements in wireless communications have reached unprecedented heights. The achievements made in wireless technology have provided pervasive network connectivity not only to the home and workplace, but also to remote areas where no wired infrastructure can reach. More recently, we have regarded this new means of communication as a primary resource and subscribe to it daily – if not constantly – in the form of mobile on-demand information services. Several wireless access technologies currently exist, such as: CDMA2000 and Wideband Code Division Multiple Access (W-CDMA) for 3G and 4G cellular communications; WiMAX (IEEE 802.16) for broadband access in metropolitan area networks (MANs); and Wi-Fi (IEEE 802.11) for wired equivalent local area networks (LANs). Nevertheless, the demand for higher data rates continues to grow, and researchers must find new ways to satisfy this need. One solution, known as multihoming, incorporates multiple network interfaces into a single device. Applied to wireless networking, multihoming can improve performance by exploiting unused resources from the radio spectrum.

Already many popular consumer electronics, like Apple's iPhone and Research in Motion's BlackBerry, come standard with Wi-Fi and cellular technologies (e.g.,

1. Part of this chapter has been published in IEEE Communications Surveys & Tutorials [1].

GSM and UMTS). Although products like these have more than one network interface, some advantages to multihoming are not being realized. Connection migration from one access technology to another, called vertical handover, is a perfect example of multihoming at its finest. Assuming a voice over IP (VoIP) telephone call is initiated within a Wi-Fi network, without a continuous series of overlapping Wi-Fi networks, the call will be dropped as soon as the phone travels even a short distance (e.g., 10s of metres). Although Wi-Fi offers high data rates at low cost, while mobile, cellular technologies can keep calls active; albeit at a higher cost with lower data rates. This is not to say these devices are not currently capable of such function, but at this time they do so only through proprietary means.

Unfortunately, our current means of guaranteeing reliability while maintaining quality control, specifically, the transmission control protocol (TCP), does not support multihoming capabilities. TCP, rather, binds a transport layer session to a single IP address; changing the IP address will kill any active session. Despite the latter, a relatively young transport layer standard called the stream control transmission protocol (SCTP), incorporates multihoming into its design. Developed by the Internet Engineering Task Force (IETF), SCTP already has a decade worth of research behind it.

It is the objective of this chapter to categorize and present the current research in SCTP as it pertains to multihomed devices with a strong interest in challenges as well as developments for the most salient issues; particularly, handover management, concurrent multipath transfer, and cross-layer activities. The rest of the chapter is organized as follows. First, we present a review of the SCTP standard, followed by an introduction to multihoming and transport layer mobility. Next, we investigate the open problems facing SCTP under different multihomed scenarios. In addition, we survey and classify the recently proposed solutions to these problems. In conclusion,

we comment on the state of this new research area while making some of our own predictions for its success.

## 2.1 Overview of the Stream Control Transmission Protocol

Developed by the IETF, SCTP [2] is a transport layer protocol that extends the functionality of the celebrated TCP standard. SCTP is a message oriented data delivery service providing full duplex connections and congestion control mechanisms. Similar to TCP, some of SCTP's features include but are not limited to: congestion and flow control, reliable data transfer, and ordered data delivery. SCTP, moreover, provides a hybrid service called partial reliable SCTP (PR-SCTP) [3], not to be confused with the user datagram protocol (UDP) and its well-known unreliable service. PR-SCTP is used to support real-time applications by relaxing retransmission guarantees [4]. Unlike TCP and UDP, however, SCTP provides support for multihomed end-points, i.e., devices with more than one network interface.

### 2.1.1 SCTP Basics

The packet definitions of contrasting transport layer protocols, SCTP and TCP, are provided in Figs. 2.1 and 2.2, respectively. Although there are many subtle differences between protocols, the most glaring is SCTP's compound packet structure. A common header, providing only basic control information, allows SCTP to differentiate packets by function. Compared to TCP's user data field, each SCTP common header is concatenated with one or more user data fields (known as chunks). Each chunk can carry either control or data information to an associated application. This

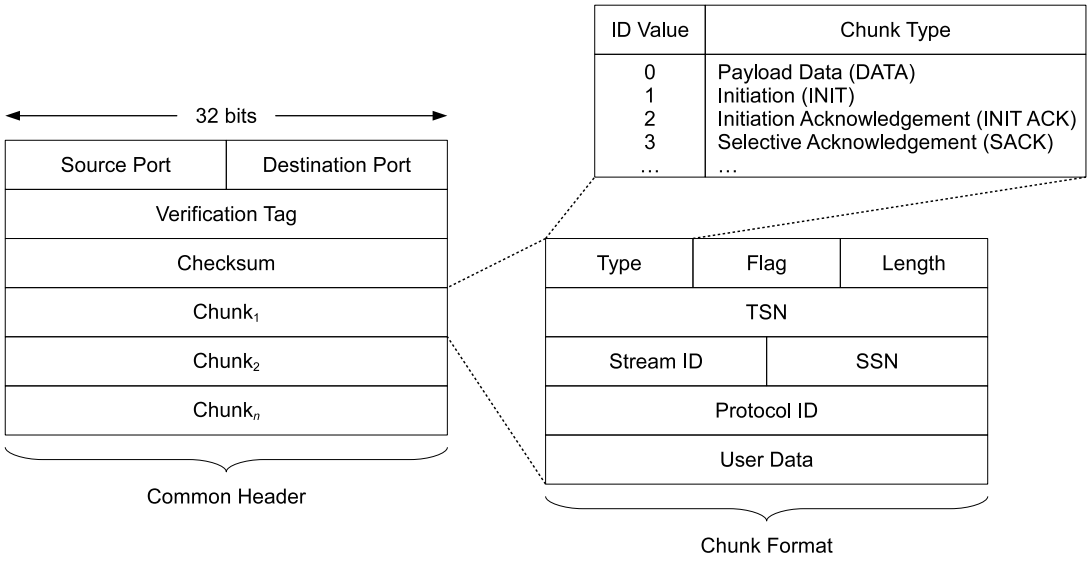


Figure 2.1: SCTP's packet format.

architecture reduces overhead and increases efficiency by bundling smaller messages together. Each chunk, moreover, classifies a packet in terms of function. Out of the possible 255 different chunk types, RFC 4960 defines 14; while another 4 are reserved by the IETF for extensions, leaving 237 available for customization.

SCTP's message oriented approach also increases utilization from the use of selective acknowledgements (SACKs). SACKs clock out reordered packets and use the available congestion window (CWND) more efficiently. TCP, on the other hand, tracks packets according to a sequential byte stream, therefore available network resources remain unused until cumulative acknowledgements (CUMACKs) are received.

Another difference is SCTP's two-tiered reliability system. At one level, transmission sequence numbers (TSNs) are used to maintain reliability and manage congestion control; while at another, stream sequence numbers (SSNs) serve as a mechanism for preserving stream independence. A stream may be allocated by an application as a means of dividing data into separate groups. For example, a web server may want to deliver multimedia data using one stream, while text uses another. If only one

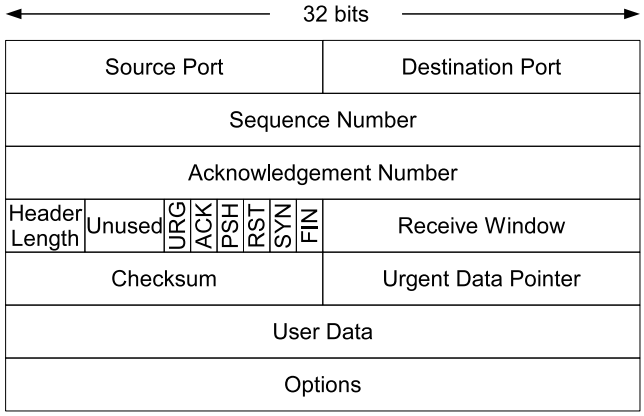


Figure 2.2: TCP's packet format.

stream is employed, as is the case with TCP, head of line (HoL) blocking can inhibit one type of data (e.g., text) while another (e.g., video) is downloading. Traditionally, the application layer has had to maintain many TCP connections, simultaneously, to avoid HoL blocking.

Turning to packet dynamics, we see that SCTP establishes associations using a four-way handshake, as opposed to TCP's three-way approach. The four-way handshake attempts to mitigate "SYN attacks", common to TCP, by replying to a client's INIT message with a cookie composed of security information only the server can verify. The client echoes the cookie and upon receipt of the final echo, a connection is established. The threat is thwarted by allocating association resources (i.e., memory) only after the initiator confirms the connection request.

During data transfer, each chunk is assigned a TSN and may be bundled with other chunks into a single packet. Messages larger than the maximum transmission unit (MTU) are fragmented into multiple data chunks and delivered separately. At the receiver, data chunks are ordered and reassembled using SSNs/TSNs and begin/end flags, respectively. In contrast to TCP's cumulative acknowledgements, SCTP sends back selective information (i.e., gap acknowledgement blocks) so congestion control

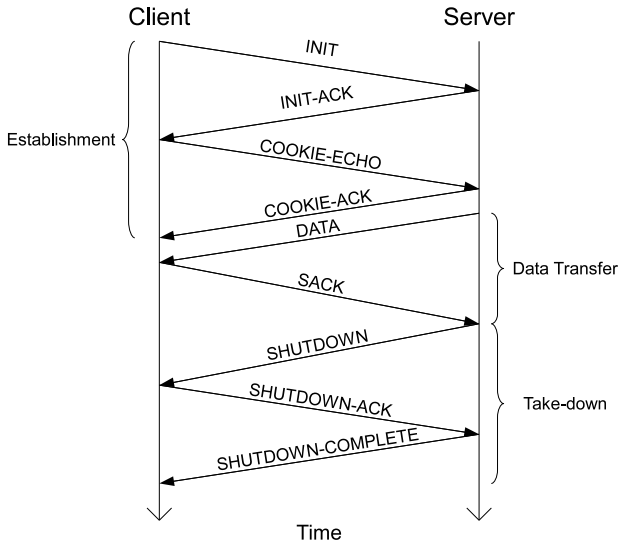


Figure 2.3: SCTP's association dynamics.

mechanisms can respond to TSNs that have been lost or received out-of-order. SACKs offer more information, albeit more bits per packet.

The dynamics of an SCTP association[2] are displayed in Fig. 2.3. This shows the establishment, data transfer, and shutdown of a normal client-server SCTP association. Additional overviews of SCTP can be found in [5–7].

### 2.1.2 SCTP Multihoming

Multihoming enables two hosts to establish a logical connection over a set of multiple network interfaces uniquely identified by separate IP addresses [5]. Typically, this type of support is applied as network-level fault tolerance or as middleware for seamless vertical handovers (i.e., switching data streams from one network technology to another without interrupting the application layer).

Currently, SCTP uses multihoming only as a backup service; when an IP address

2. In terms of SCTP, the word association is analogous to a TCP connection.

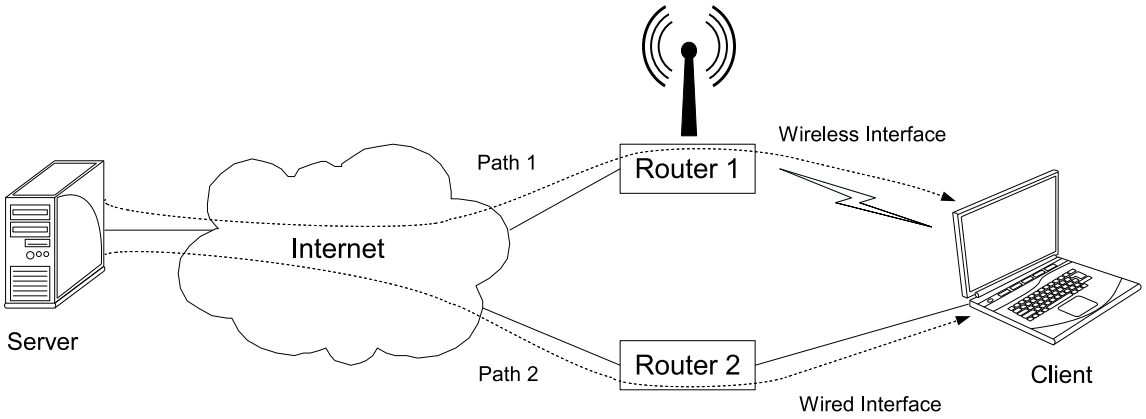


Figure 2.4: A multihoming scenario.

becomes unreachable, SCTP will attempt to recover communication by sending new data to a secondary IP address. SCTP maintains an active list of alternate destination addresses by either sending periodic *heartbeat* probes or receiving acknowledgements from retransmitted data[3].

As previously stated, SCTP provides mechanisms for both flow and congestion control; flow control is on a per association basis, and congestion control is managed per IP address. This is necessary to support multihomed interfaces with varying performance capabilities. Fig. 2.4 illustrates a typical sender/receiver pair in a multihomed environment using two different network interface technologies (e.g., Wi-Fi and Ethernet). Although the image depicts the sender with only one interface, this is not always the case. For example, it is possible for two multihomed end-points to use SCTP for voice communication; while in transit, each end-point may switch access technologies any number of times.

3. SCTP can employ a retransmission policy [8], e.g., one that retransmits lost packets to an alternate destination.

### 2.1.3 SCTP Mobility

For a long time the Internet was uniform and static in nature, so legacy transport layer protocols, like TCP and UDP, never really had mobility in mind. Today, however, networks thrive on diversity and have evolved with an ability to alter their topological representation on-demand. Unfortunately, even with the development of SCTP, mobility was not a functional requirement. Not only must the transport layer negotiate available network resources for the application layer, it also needs to support network layer dynamic reconfiguration capabilities. Network reconfiguration occurs often when wireless devices move from one base station to another, or sometimes even when a link fails (e.g., cut, unplugged, overly congested). Although much of these adjustments may remain hidden from the transport layer, if the IP address changes, the application will lose its current session.

The popular choice for network mobility has been the Mobile IP (MIP) standard [9]. Unfortunately, a major drawback of MIP is that it does not support connection migration at the transport layer, even though many situations find this limitation undesirable: for example, an Internet service provider (ISP) updates a host's IP address; a new plug and play (PnP) network card is added to a multihomed server; or more commonly, a mobile user moves from one subnet to another. In each of these scenarios, a service disruption would pause communication while the current association was torn down and reconfigured.

Although various research efforts have explored the area of transport layer mobility [10], dynamic address reconfiguration (DAR) [11] is a simple yet practical solution to SCTP mobility. DAR provides an SCTP association with support for three new features: (1) dynamic addition of IP addresses, (2) dynamic deletion of IP addresses, and (3) primary address update requests. The extension also defines two new

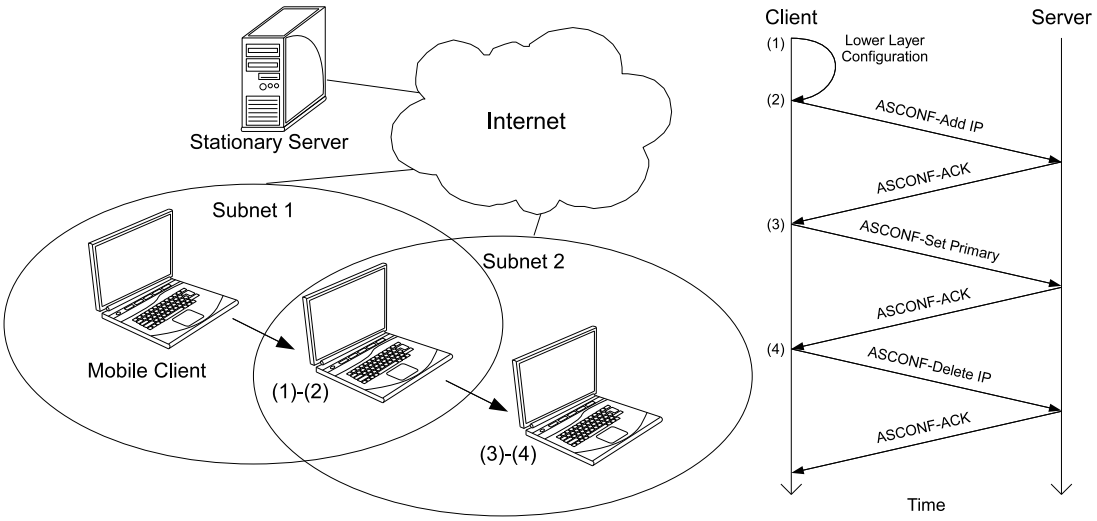


Figure 2.5: Simple multihoming example using DAR.

chunk types, address configuration change chunk (ASCONF) and address configuration acknowledgment chunk (ASCONF-ACK). While the ASCONF chunks are used to either add/delete IP addresses or change the primary path, ASCONF-ACK chunks simply respond with success or failure messages to ASCONF requests.

A simple example of a mobile client associated with a static server across the Internet is shown in Fig. 2.5. The figure illustrates a mobile client moving from one subnet to another and the necessary DAR messages exchanged between both endpoints to keep the connection alive. At time (1), the mobile establishes a network presence in subnet 2, then informs the server of its additional IP address at time (2). Next, the primary path is updated to the new IP address, while the older one is removed from the association, (3) - (4). Although this is an intuitive solution for the transport layer, the lower layers (e.g., network, link, physical) may play a more complex negotiation game before any ASCONF chunks are transmitted.

DAR, together with SCTP, is commonly referred to as mSCTP (or mobile SCTP). Several other variations of mSCTP include: cSCTP, SIGMA, and mSCTP+

[12–14]. It should be noted, though, that no version of mSCTP offers a completely decentralized system. For example, to initiate communication with a mobile that continually changes its address, an end-point needs to know where to look first. The location manager, like the name suggests, is a network entity that locates mobile clients. The location manager is statically stationed and keeps real-time address information on behalf of its mobile client, but is not necessarily bound to any particular network its corresponding client may be attached to.

## 2.2 Problems, Issues, and Challenges

The purpose of this section is to brief the reader on the most salient issues affecting SCTP multihoming, that is, handover management, concurrent multipath transfer, and cross-layer activities. In addition sub-problems related to these issues are introduced and their importance explained. Later in Section 2.3, a review of solutions to these sub-problems (proposed in the literature) is presented.

### 2.2.1 Handover Management

Handover offers the wireless client mobility by keeping connections alive while traversing access points. Multihoming expands that mobility by adding a wider range of access networks. To understand the role of handover in transport layer multihoming, we have created three new subcategories: (1) preemptive path selection, (2) fault tolerant path selection, and (3) post handover synchronization. In each subcategory, we explore an integral problem inherent to handovers at the transport layer.

#### 2.2.1.1 Preemptive Path Selection

A typical multihomed environment will present the transport layer with a number of destination paths with disparate performance capabilities. Among these paths, SCTP must choose one for communication. Often, an end-point will want to choose the "best" path; providing either best-effort or quality of service (QoS) guarantees. Usually metrics like bandwidth, delay, and jitter are adequate, but if the multihomed environment consists of wireless links, signal strength and loss rate are also critical. Given the range of networking applications, e.g., instant messages and video chat to name a few, and including other constraints, such as usage fees and power consumption, the "best" will not always be a *one size fits all*. Clearly then, multihoming increases handover complexity, by introducing a range of decision problems with a multitude of variables to explore.

#### 2.2.1.2 Fault Tolerant Path Selection

SCTP avoids the path selection problem by letting a destination become unreachable (i.e., fail) before choosing an alternate. Without direct access to the link layer, however, SCTP has no way of knowing whether a fault has occurred. For example, consider a client end-point downloading a file using one of two interfaces. If the client's active interface loses its gateway connectivity (e.g., moves out of range of a base station), the server end-point will continue sending data – unknowingly – to a broken link. Notifying the server of connection loss (through the secondary interface) is trivial, but what happens when the mobile is slowly losing connectivity? For example, the wireless link is active but packet losses are increasing. Furthermore, what if the active interface already offers the highest data rate? Should the transport layer switch to a slower rate for better stability, or stay with its current incumbent for

higher throughput potential? It is complicated questions like these that are the main reason why SCTP relies solely on fault tolerance for handover support. Nevertheless, multihoming should make losses more avoidable and interruption delays nonexistent.

#### 2.2.1.3 Post Handover Synchronization

Whether handover is invoked from proactive (preemptive) or reactive (fault tolerant) path selection, following transition the transport layer is still susceptible to inefficiencies. *Spurious retransmissions*, brought on by packet reordering, causes the transmission rate to drop significantly. Reordering typically occurs from packets being delayed along a slow path while others are ushered ahead over a faster one. An undesired consequence of reordering is illegitimate loss indications prompting CWND cutbacks. Without direct access to the link layer, the transport layer blindly infers packet loss by evaluating the order of arrivals. If too many packets are received out-of-order, the transport layer assumes loss and makes concessions to expedite recovery.

### 2.2.2 Concurrent Multipath Transfer

As mobile users manoeuvre through heterogeneous access networks, at times they may be presented with more than one access point with disparate performance capabilities and economic costs. For example, while one network may provide a large coverage area over one that is smaller, it may also bare higher prices. Furthermore, power consumption may also reduce battery life depending on base station proximity or media access control (MAC) specifications. But if these caveats are of no concern, playing some sort of complex network selection game will only add overhead. Interestingly enough, however, when a device is multihomed, it is theoretically possible to use multiple network technologies simultaneously to aggregate throughput poten-

tial. The research community currently refers to such simultaneous transmissions as concurrent multipath transfer, but has also used terms like bandwidth aggregation, resource pooling, inverse multiplexing, load sharing, and even striping in similar contexts. Generally speaking, if more network resources (i.e., communication channels) were present, one would naturally presume an increase in upload speeds. Unfortunately, this has not been the case with SCTP for a number reasons: (1) unnecessary fast retransmissions; (2) crippled congestion window growth; (3) superfluous network traffic; (4) receive buffer blocking; and (5) naive scheduling.

#### 2.2.2.1 Unnecessary Fast Retransmissions

Fast retransmission causes the sender to interpret network congestion, thereby reducing its CWND. Moreover, a SACK with a gap report does not necessarily imply packet loss – that is to say, missing packets could be delayed and thus reordered. Still, SCTP infers a packet to be lost from additional SACKs because the probability of loss increases with every gap report received. Even in a lossless system, however, CMT reorders packets on a constant basis – diminishing connection quality in the absence of loss.

#### 2.2.2.2 Crippled Congestion Window Growth

Another side-effect of CMT is overly conservative congestion window growth. When SACKs are received by the sender, they contain only new gap reports but no CUMACKs. Since it is the CUMACKs which grow the CWND, previous gap reports that have already acknowledged packets will have no impact on how the CWND grows. Even though packets may have arrived in-order, this behaviour will limit CWND growth and prevent bursts of new data.

#### 2.2.2.3 Superfluous Network Traffic

The next inefficiency caused by CMT is in regard to additional network traffic. Both SCTP and TCP reduce acknowledgement traffic by delaying acknowledgements until at least two can be sent together. Packets that are received out-of-order, rather, should send a duplicate acknowledgement (i.e., the last CUMACK) immediately [2, 15]. Since packets are frequently reordered during CMT, acknowledgement traffic increases gratuitously.

#### 2.2.2.4 Receive Buffer Blocking

Realistically we cannot assume a receiver will have infinite buffer space. In fact, mobile devices have very limited memory to begin with. Constrained buffers pose an even greater problem if different paths have disparate bandwidth characteristics. A sender is allowed to send only when the receive window (RWND), i.e., the amount of free buffer space, is greater than zero. Furthermore, the RWND increases only when packets are received in-order and passed to the application layer. Again, since packets are regularly reordered during CMT, new transmissions can be temporarily blocked. When the RWND is full, blocking is devastating because it lowers network utilization and ultimately disables throughput.

#### 2.2.2.5 Naive Scheduling

Simultaneous transmissions over paths with disparate bandwidth characteristics will not yield synchronous receptions. This environment will hinder the performance of a protocol offering ordered data delivery because higher sequence numbers will be received before lower ones. Furthermore, a simple round robin approach to scheduling

packet transmissions over multiple paths will undoubtedly lower throughput, primarily because out-of-order arrivals will be queued at the receive buffer (RBUF). Even in the absence of loss or a constrained buffer, CMT needs intelligent scheduling to increase throughput and lower receiver-side queueing.

### 2.2.3 Cross-layer Activities

This last area relates the preceding subsections by looking more closely at some of the additional responsibilities multihoming has imposed on the transport layer.

#### 2.2.3.1 Bandwidth Estimation

Choosing the "best" path usually requires measuring bandwidth, delay, or jitter to provide best-effort or even QoS guarantees; how to measure bandwidth, delay, or jitter is an entire problem unto its own. Like TCP, SCTP will find difficulty in gathering accurate path measurements since it lies, detached from intermediary physical links, at the end-points of a connection. Moreover, if a metric can be gauged there is no guarantee it will remain accurate for any length of time since a network path is shared among many different sources with unknown traffic patterns. A conflict of interest can also arise if the transport layer has to send too much traffic into the network just to determine capacity. Each SCTP source needs to manage its own traffic efficiently so as not to overwhelm the entire network. While it may use the continuous flow of acknowledgements to infer bandwidth over the primary path, currently SCTP can only measure the capacity of a secondary path from periodic probing.

#### 2.2.3.2 Wireless Error Notification

When the network path is composed of wireless and wired links, the cause of each loss is more difficult to discern. It is common practise to assume wired links will not experience transmission errors, but may lose packets due to buffer overflows. On the other hand, while buffering is still a problem for wireless domains, the effects of signal fading, interference, noise, or Doppler shifts can corrupt packet transmissions. Since transport layer protocols are designed only to handle network congestion, losses over wireless channels are ambiguous. Due to this ambiguity, an end-point may not be able to discern congestion from wireless losses, leading to mismanaged resources and poor performance.

#### 2.2.3.3 Network Intelligence

Routing is usually reserved for network layer protocols like Intermediate System to Intermediate System (IS-IS) and Open Shortest Path First (OSPF). Assuming the system's terminating point is its IP address, protocols like IS-IS and OSPF work fine; but if the actual terminating point lies beyond the IP address, like in a multihomed system, some inefficiencies may go undetected. While the network layer sees a system of independent links between IP addresses, the multihomed transport layer sees only IP systems (i.e., pairs of IP addresses), where each IP system may or may not share common links. This raises issues of ambiguity, and challenges the notion that each IP system is truly independent. Similar to path selection, found in the handover management problem, the multihomed transport layer will have additional routing responsibilities, regardless of mobility.

## 2.3 Solutions, Strategies, and Techniques

This section highlights the research efforts used to address the main issues surrounding SCTP multihoming. Note that each solution approach corresponds to a sub-problem with the same heading introduced in Section 2.2.

### 2.3.1 Handover Management

#### 2.3.1.1 Preemptive Path Selection

Making handover decisions or selecting an access network is a complicated problem. Many sophisticated strategies, such as economic cost functions or even machine learning techniques like fuzzy logic and neural networks have been used extensively to make optimized handover decisions. In fact, much of the work in this area has already been reported. For a comprehensive survey of vertical handover decision algorithms, we refer the interested reader to [16]. Vertical handovers involve reconfiguring an endpoint's ingress access from one network technology to another. In [16], the handover problem primarily focuses on choosing the "best" access network given a set of user constraints. When comparing algorithms, handover performance can be evaluated by delay, number of handovers, handover failure probability, and throughput. We also direct the reader to [17], another survey that offers supplemental material on handover classification and mechanics.

Path selection strategies not mentioned in previous surveys include: SIGMA and ECHO. SIGMA, or Seamless IP diveristy based Generalized Mobility Architecture [13], is actually a mSCTP framework. SIGMA offers mobility to multihomed wireless devices through transport layer handover, location management, and simple path optimization. We say simple because the path selection tool compares only one variable, that is, received signal strength (RSS). When it comes to path selection,

however, SIGMA is proactive; instead of a threshold, handover is initiated every time an alternate interface has better signal strength.

Alternatively, ECHO, or endpoint centric handover [18], uses a more sophisticated path selection strategy. ECHO's primary objective, is to offer QoS guarantees for VoIP calls using PR-SCTP. ECHO employs the E-model [19], an International Telecommunication Union (ITU) planning tool, to estimate the quality of end-to-end connections. Based on subjective testing, the E-Model rates the quality of a connection on a scale from 0 to 100. The technique transforms individual transmission parameters (e.g., signal strength, delay, jitter) into different *impairment factors* and adds them together. The output of the E-Model, $R$, is calculated by

$$R = R_o - I_s - I_d - I_e + A, \tag{2.1}$$

where $R_o$ is the basic signal-to-noise ratio (SNR), $I_s$ represents impairments that occur simultaneously with voice encoding, $I_d$ sums all impairments due to delay, $I_e$ gives the effects of equipment, and $A$ is an advantage factor. The last variable, $A$, compensates for certain impairment factors by attempting to *level the playing field* (e.g., a willingness to accept lower quality connections under wireless conditions). Through a series of assumptions and simplifications, ECHO reduces the E-Model to

$$R = 93.34 - I_d - I_e, \tag{2.2}$$

where the terms $I_d$ and $I_e$ then correspond, more specifically, to delay/jitter and loss, respectively. ECHO regularly measures $I_d$ and $I_e$ by transmitting the same data over each available path; then chooses the path with the best $R$ score.

#### 2.3.1.2 Fault Tolerant Path Selection

By foregoing the complicated phase of preemptive path selection, an easier way to make handover decisions is to simply wait until it is absolutely necessary, particularly, when the network tells you to. There are two major problems with this logic: (1) communication interruption, and (2) avoidable losses. The first problem occurs from simply waiting until it is absolutely necessary to update the transmission path. If we have to react to a problem, we will undoubtedly incur some kind of delay. But we can exacerbate that delay by choosing the wrong end-point to manage faults. For instance, the sender uses a parameter called *path.max.retrans* (PMR) to determine a destination's reachability. Although failover (i.e., handover due to failure) is considered an implementation specific issue [2], the current recommendation is to compare the number of consecutive failed retransmission attempts to PMR. Then if this number exceeds PMR, the primary path should be considered inactive and an alternative will be selected. Since the retransmission timeout (RTO) increases exponentially following each consecutive loss, the total time necessary to detect a path failure can be expressed as $\sum_{i=0}^{\mathrm{PMR}} 2^i \mathrm{RTO}$. RFC 4960 suggests the following default settings: $\mathrm{RTO}_{\mathrm{min}} = 1$ s, $\mathrm{RTO}_{\mathrm{max}} = 60$ s, and PMR = 5. An SCTP association can then expect to experience a failover delay (i.e. the time from a link failure until a primary path update) to be anywhere between 63 and 360 seconds.

Staying with sender-side failure detection, some studies have tried to mitigate failover delay by monitoring the nature of loss events. The scheme proposed in [20], sending-buffer multicast-aided retransmission with fast retransmission (SMART-FRX), works in three ways. The first is a minor change to the retransmission policy; lost packets notified by missing reports are always retransmitted to the same address, and those that have timed out are sent to an alternate destination. This is attributed

to the assumption that losses from missing reports are due to network errors rather than poor conditions; if an alternate path has lower bandwidth over the primary path, a delay due to retransmission may inadvertently reduce throughput even further. The second method avoids handover loss by simultaneously transmitting the same data over multiple paths during periods of instability. Finally, the third approach mitigates failover delay by monitoring loss events. If at some point the number of consecutive retransmissions on the primary path exceeds PMR, the primary path becomes inactive and an alternate takes over. If, however, a retransmission over the primary path is successful before exceeding PMR, no change will take place.

The multipath transmission algorithm (MTA) is a method aimed at reducing losses while the conditions of the primary path begin degrade [21]. Avoiding any CWND and reordering problems, the authors direct their work toward real-time applications and simply disable the congestion avoidance and fast retransmission. Similar to [20], MTA also transmits the same data over multiple paths during periods of instability. The algorithm implements two transmission modes: *singlepath* and *multipath*. During singlepath, the source sends information only along the primary path, while multiple copies of the same information are sent along an alternate path during multipath mode. Multipath mode starts when the number of consecutive retransmissions to the primary path reaches Multipath.Threshold (MT), such that MT $<$ PMR. In multipath mode, the sender monitors the stability of the new path by sending heartbeat packets. When the sender receives two consecutive acknowledgements for heartbeat packets, a new counter is increased by one and compared to a parameter called Stability.Threshold (ST). If this new stability counter exceeds ST before the number of consecutive retransmissions overtakes PMR, the alternate path will become the new primary path.

The two most basic operations required for transport layer handover are: (1)

adding a new IP address, and (2) changing destination paths. When to exploit these operations now becomes the quintessential question. An experiment from [22] describes a dualhomed SCTP receiver crossing randomly between two access points. At the same time, a file is being downloaded from a stationary server across the Internet; transport layer *Add-IP* and *Path-Change* functions are triggered when preset RSS thresholds are broken. Analysis of this study focused on the effect of signal strength for each threshold. For example, both aggressive and conservative thresholds were chosen for either Add-IP or Path-Change operations, where the terms aggressive and conservative refer to lower and higher threshold levels, respectively. In terms of Add-IP, the aggressive rule adds a new IP address when the signal strength reaches some minimum threshold. On the other hand, the conservative rule only adds a new IP when the strength is greater than that of the current IP address. The results of the experiment showed using an aggressive Add-IP with a conservative Path-Change will yield the fastest download times. Clearly, download times will be faster under the aggressive Add-IP rule since the alternate path is available sooner. But higher throughput from the conservative Path-Change rule may not be as apparent. It was said that by making frequent path changes, as with the aggressive rule, the association becomes more unstable, hence lower performance. With respect to this type of scenario, a hysteresis threshold (i.e., introduced state-change memory) could mitigate undesirable connection oscillation between access networks [23, 24].

Before moving on, we would like to point out an intrinsic vertical handover problem. Again, assume two end-points share an SCTP association across the Internet. Currently, the wireless multihomed receiving end-point finds itself moving from one base station to another and between subnets. Although the primary path may use either of the two available destination IP addresses while the receiving end-point is in range of both base stations, when the signal becomes so low that a connection cannot

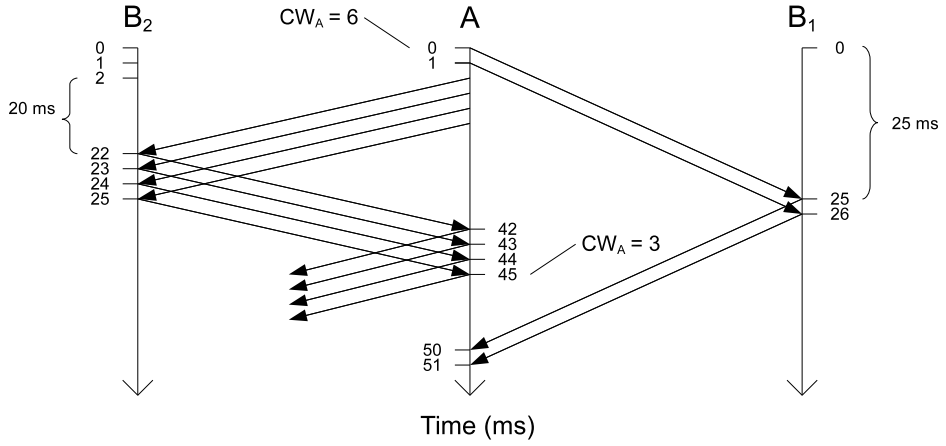


Figure 2.6: Congestion window cutbacks and spurious retransmissions during handover.

be maintained, the primary path of the SCTP association will have to be updated to acknowledge the receiving end-point's only location of network access. The question then is, "At what point should the primary path be updated?" While analyzing this problem, we may first want to define what our main goal is, e.g., is it more important to maximize throughput to transfer data as fast as possible or perhaps minimize loss so that a stable connection is retained during handover? We then might be interested in such things as bandwidth (i.e., data rate) and whether the receiver is entering a subnet where the bandwidth will be higher or lower compared to the old subnet. In either event, the state variables controlling SCTP will need to be monitored carefully.

#### 2.3.1.3 Post Handover Synchronization

Network capabilities will often change after handover; sometimes for the better and other times for the worse. Regardless of the outcome, synchronization issues at the transport layer can lead to undesirable results. A handover example describing the effect of reordered packets is shown in Fig. 2.6. In this example end-point A is sending to multihomed end-point B using address 1 (i.e., $B_1$). The propagation delay from A

to $B_1$ is 25 ms, where all other delays (i.e., transmission, queueing, and processing) are considered negligible. Furthermore, the one way delay from A to address 2 of B (i.e., $B_2$) is 20 ms. At time 0 the CWND is 6 packets. After sending two packets, and at time 2, the primary path is updated and the remaining four packets are sent to $B_2$. Since it takes longer for the first two packets to reach B over $B_1$ than it does the remaining four over $B_2$, when the acknowledgements for the last four reach A, the fourth missing report for the first two packets will trigger a fast recovery due to the higher than expected sequence numbers. This causes the CWND to be halved and retransmissions made at time 45, albeit unnecessary, since the first two packets will be acknowledged at times 50 and 51, respectively. Following the loss indication at time 47, the spurious retransmissions made by the sender only add to the problem by further congesting the network with redundant data.

Ideally, we would like to avoid illegitimate loss indications and do away with spurious retransmissions altogether, but finding a solution is no easy task. Currently, techniques work only on the principle of spurious retransmission detection. For example, the Eifel algorithm [25] informs a sender of premature retransmissions after receiving acknowledgements for packets sent before retransmissions. Following a loss indication and retransmission of any missing packets, the Eifel algorithm simply compares the transmission time of all acknowledged packets received before the retransmitted ones. If an acknowledgement is received for a packet with a transmission time before the loss indication, a spurious retransmission is detected and congestion control is reverted back to its previous state.

Although the simplicity of the Eifel algorithm can mitigate some performance degradation due to spurious retransmissions with prompt detection, it suffers from an inability to differentiate between reordering and actual congestion loss. An example of such is shown in Fig. 2.7. The example shows a delayed packet transmitted at

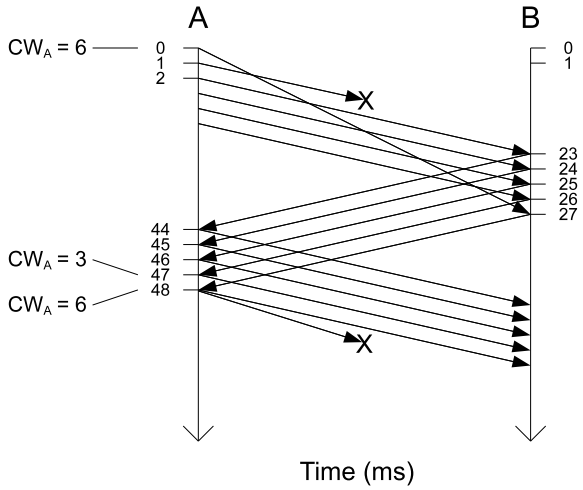


Figure 2.7: The Eifel problem.

time 0 followed by a lost packet at time 1. When the fourth missing report arrives at time 47 a retransmission is made on behalf of the first packet, albeit at time 48 an acknowledgement for the first packet arrives. Depending on the retroactive policy for spurious detection, using the Eifel algorithm could return SCTP to its original congestion control state, which may result in another loss due to congestion, all the while disregarding the fact that the second packet was actually lost.

Another strategy, known as duplicate SACKs (DSACKs) [26], will wait for all retransmissions to be acknowledged before concluding whether any or all were indeed spurious. But again the system must make spurious retransmissions before determining if they were warranted or not. This can be likened to the idea of *throwing an alleged witch into a lake only to see if she floats*. Ladha et al. propose the combination of Eifel and DSACK to combat spurious retransmissions due to reordering in [27]. Fig. 2.8 provides a modified flowchart of the combined process highlighting the necessary steps to determine spurious retransmissions with respect to the functional requirements proposed in [27]. Note that by testing whether a CUMACK is greater or equal to the highest retransmitted packet implicitly concludes that the original

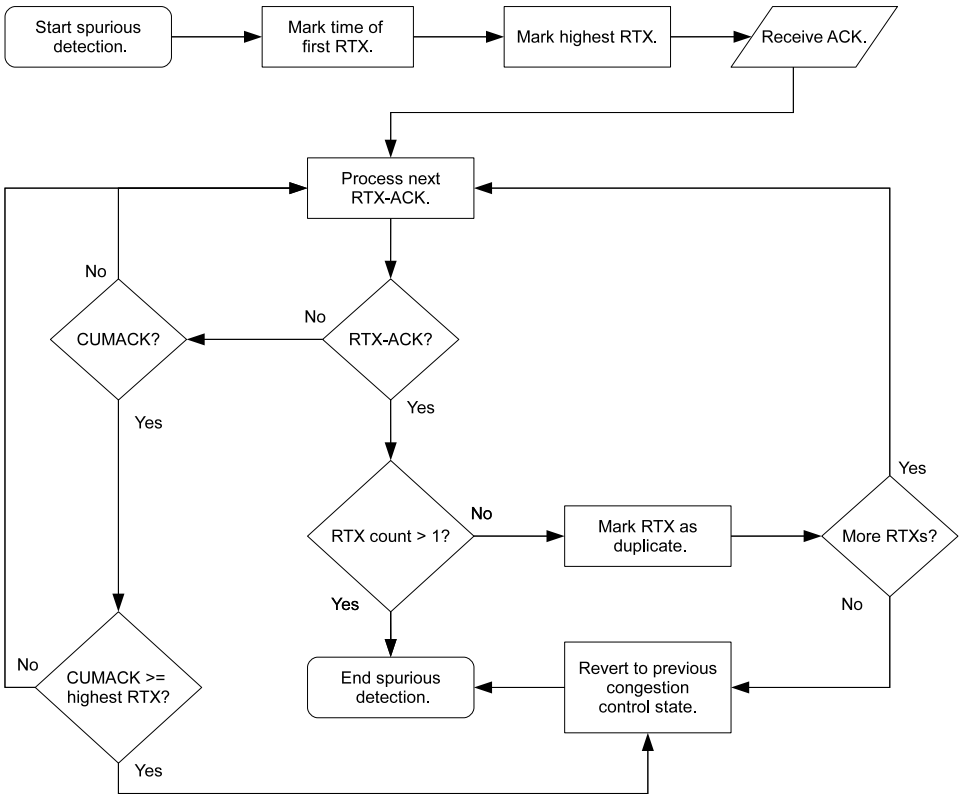


Figure 2.8: Eifel/DSACK algorithm.

transmission time is also less than the send time of the first retransmission.

Presented in [28], SCTP handover optimization (SHOP) is a method for reducing packet reordering after handover. SHOP is a proactive scheme that monitors the average RTT of any impending handover path. If the new path has a lower latency over the old one (i.e., $\Delta\text{RTT} > 0$), then following handover all queued transmissions are immediately postponed. SHOP simply backs off for a period of $\delta \cdot \Delta\text{RTT}$ until ordered data arrivals can be better approximated. The coefficient $\delta$ is a connection parameter applied to avoid inaccurate estimations due to receiver-side delayed acknowledgements.

A final proposal called handover retransmission for mSCTP (HR-mSCTP) is presented in [29]. The proposed work also sends redundant packets to prevent CWND

cuts due to illegitimate loss indications, or timeouts following handover. By preemptively retransmitting all unacknowledged packets before sending new ones along the alternate path, the authors guarantee the CWND will not be affected by spurious retransmissions.

### 2.3.2 Concurrent Multipath Transfer

#### 2.3.2.1 Unnecessary Fast Retransmissions

Iyengar et al. solve the spurious retransmission problem for CMT with their Split Fast Retransmit (SFR) algorithm [30]. SFR validates the legitimacy of missing reports by analyzing the destination address of reordered packets; specifically, SFR tracks the highest acknowledged TSN for each destination address, indicated by the variable *highest*. As an example, if TSNs 1 through 5 were sent to destination 1 and TSNs 6 through 10 were sent to destination 2 and the receiving SACK carried TSN acknowledgements 4 and 6; then *highest* for destination 1 would be 4, while *highest* for destination 2 would be 6. We have provided a simplified version of the SFR algorithm in Fig. 2.9.

#### 2.3.2.2 Crippled Congestion Window Growth

Another algorithm in [30], Congestion Window Update for CMT (CUC), allows the CWND of each destination to grow independently. Instead of using a single CUMACK variable, CUC employs individual CUMACKs for each destination. After each transmission, CUC implicitly notes the expected order of TSNs for each destination address. Again, if TSNs 1 through 5 were sent to destination 1 and TSNs 6 through 10 were sent to destination 2 and the receiving SACK carried TSN acknowledgements 4

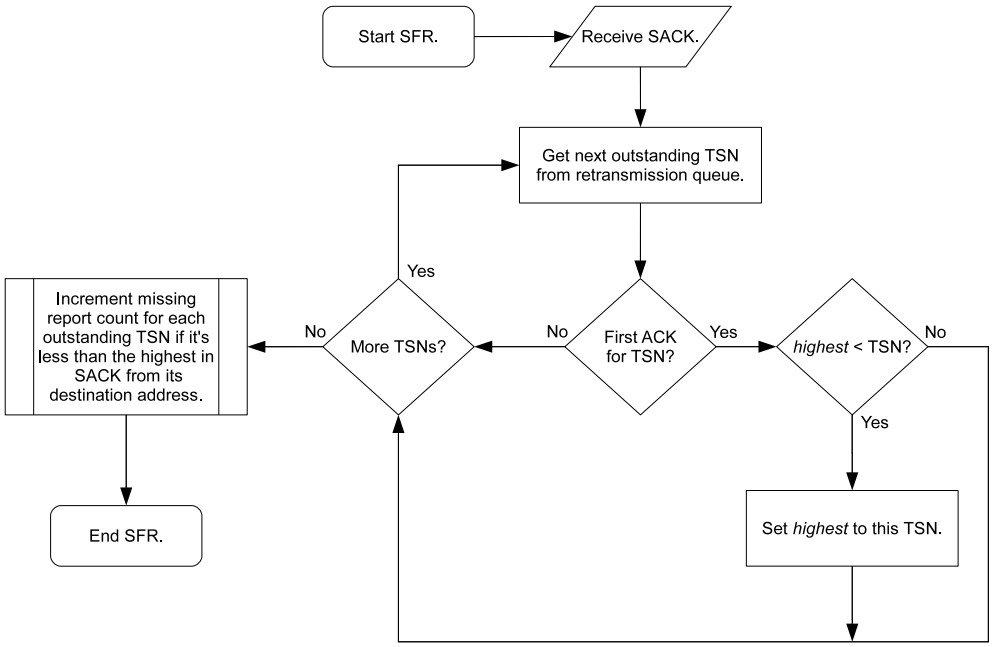


Figure 2.9: The Split Fast Retransmit Algorithm.

and 6; then, a CUMACK would update the congestion on destination 2 but not on 1. An interpretation of CUC is provided in Fig. 2.10.

To be complete, we should also mention the independent per path congestion control for SCTP (IPCC-SCTP) algorithm [31] as well as Load-Sharing-SCTP (LS-SCTP) [32]. IPCC-SCTP and LS-SCTP will also prevent spurious retransmissions and poor window growth by assigning each TSN a unique path sequence number (PSN). Upon reception of a SACK, they perform a similar operation as SFR by marking only those TSNs as received or missing if their corresponding PSNs are either in or out-of-order, respectively. Still, IPCC-SCTP incorporates an implicit extension to the sender while adhering to the SCTP standard; LS-SCTP requires changes to both the sender and receiver as well as reformation of the SCTP packet.

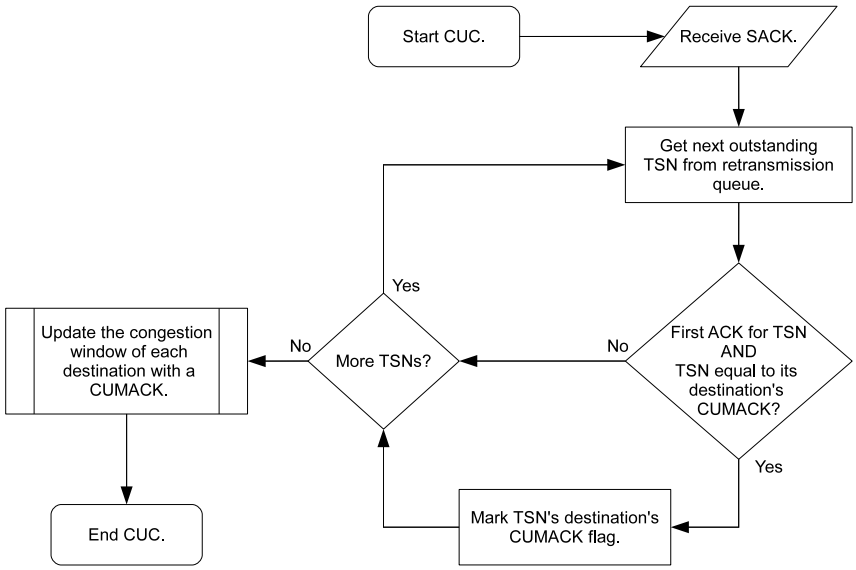


Figure 2.10: Congestion Window Update for CMT.

#### 2.3.2.3 Superfluous Network Traffic

To curb the barrage of redundant acknowledgements from reordered packets, a routine called Delayed Acknowledgement for CMT (DAC) was incorporated into SFR [30]. DAC first mandates a delayed acknowledgement rule regardless of gaps. Then if a SACK acknowledges TSNs sent to the same destination as one from the retransmission queue, it should increment its missing report count by the number of packets received prior to the previous SACK, i.e., 1 or 2 packets.

#### 2.3.2.4 Receive Buffer Blocking

The performance of CMT when constrained by the RBUF was showcased in [33, 34]. The authors demonstrated poor transfer times using a shared RBUF between high and low bandwidth paths. In fact, in some circumstances using only the higher bandwidth path provided better performance results. The problem was dubbed *receive buffer blocking*; described as an inefficiency causing a sender to pause transmission while cumulative packets remain unacknowledged. Attempts to mitigate this problem took

the form in various retransmission policies, such as: (1) send to the same destination after a loss; (2) send to any destination with an open[4] CWND; (3) send to the destination with the largest CWND; (4) send to the destination with the largest slow-start threshold (SSTHRSH); and (5) send to the destination with the lowest loss rate. Although conclusions were vague, the authors felt loss rate was of most importance when choosing a destination for retransmission. A final strategy, also based on retransmission policy, suggests the combination of retransmission policies (2) - (4) when path characteristics are similar [35]. For instance, when multiple CWNDs are equal, choose a destination based on SSTHRSH or loss rate; else if all variables are the same, make a random selection.

A similar blocking effect can also occur at the sender if acknowledged packets remain in the send buffer (SBUF) for too long; even if packets are received successfully, they must remain in the SBUF until an acknowledgement guarantees ordered delivery. Keeping packets in the SBUF, even after they have been acknowledged, is a safe guard against receiver-side reneging[5]. Assuming a receiver never reneges, Yilmaz et al. [36] proposed the *non-renegable* selective acknowledgement (NR-SACK) as a way of freeing up room for new transmissions. The new acknowledgement type simply tells the sender to remove acknowledged packets from the SBUF, regardless of reordering.

The receive buffer blocking problem can also be extended to path failures. Assuming the presence of more than one destination and following a timeout, the sender can avoid a second consecutive loss by retransmitting to an alternate destination. Even though this can mitigate the initial damage with prompt recovery, if a reactive failover scheme is employed, the sender will continue transmitting new data to a bro-

4. The term *open* means the size of the CWND is greater than the number of outstanding packets.

5. A receiver's ability to give an acknowledgement only to disregard it later.

ken link until PMR reaches its threshold. During each failed attempt, receive buffer blocking will degrade connection quality. Moreover, if the RTO is exponentially increased after each consecutive loss, the blocking problem will suffer a similar delay. Noted in [37], this problem is tackled by introducing a new state called *potentially-failed* (PF). Following a timeout, the corresponding destination will be flagged PF. All new data is then transmitted over an alternate path and a heartbeat probe sent to the PF destination. If and only if a heartbeat is acknowledged will the PF destination be returned to a state of congestion avoidance.

More recently, modelling techniques were applied to CMT in an attempt to avoid receive buffer blocking altogether. In [38], Yang et al. modify the well known PFTK model [39] to calculate a minimum RBUF size, such that receive buffer blocking cannot exist. The same authors, moreover, used a similar model in [40], to choose the configuration of destination addresses that will maximize throughput for a given RBUF size.

#### 2.3.2.5 Naive Scheduling

Although retransmission policy can offer some relief, we think intelligent scheduling shows the most promise for tackling the receive buffer blocking problem. Using performance characteristics (e.g., bandwidth and delay), an ideal scheduler should be able to approximate packet delivery times; thus minimizing reordering at the receiver. The first scheduling algorithm used for CMT, however, simply transmitted the next cumulative packet to the first destination with an open CWND [30], regardless of delivery time. As this method ignores bandwidth and delay, it has no effect on receive buffer blocking. Such an approach will be referred to as *naive*, since no intelligence is used in the scheduling process.

Contrary to the naive approach, [41] proposed the bandwidth aware scheduler (BAS) to approximate packet delivery times before transmission. BAS assigns newly created packets to virtual send queues corresponding to one of the receiver's destination addresses. To make its scheduling decision, BAS uses a destination's bandwidth estimate as well as a backlog of scheduled packets to predict delivery times. A new packet is then placed in the send queue of the destination with the earliest delivery time. BAS calculates a packet's delivery time by

$$C_i = C_i + \frac{D}{B_i} + P_i, \tag{2.3}$$

where $C_i$ is the current delivery time of packets on path $i$, $D$ is the packet size, $B_i$ is the bandwidth (i.e., data rate) on path $i$, and $P_i$ is the propagation delay. Although BAS decides which packet is sent to which destination, it does not control when packets are transmitted.

Unfortunately, maintaining an accurate backlog can be difficult, especially if packets are lost and need to be retransmitted. To simplify the matter, the same authors improved BAS in [4], but this time focusing only on real-time data transfer. Employing an adaptive scheduling algorithm, the authors proposed a new transport protocol called Westwood SCTP with Partial Reliability (W-PR-SCTP). Using a similar method to TCP Westwood+ [42], W-PR-SCTP bases its path estimates on contiguous, non-overlapping time windows, taking the maximum of either one RTT, or 50 ms. Bandwidth is then measured using a smooth auto regressive moving average. Data chunks, moreover, are scheduled and assigned according to a path's *reception index*. The reception index of a path is calculated by dividing the accumulated size of the current chunk, any outstanding chunk(s), and any buffered chunk(s) by a bandwidth estimate. Although W-PR-SCTP ignores the size of the RWND by assuming

an infinite RBUF, simulation studies showed that W-PR-SCTP would outperform a naive or greedy scheduler (i.e., one that sends data over a path as soon as the CWND is available) in terms of jitter and overall throughput.

Lastly, an optimal scheduler was proposed in [43]. The authors first model the problem of distributing data packets over multiple paths as a Markov chain. Then a Markov decision process (MDP) is formulated to specify a scheduling policy, specifically, which actions are taken given system state and time step. Following this, the On-line Policy Iteration (OPI) algorithm was proposed to approximate optimality. Although this work has substantial merit, it forgets two major constraints of the transport layer, that is, limited receive buffer and ordered data delivery. Without mention of these constraints, optimal throughput could be achieved, but unrealistically.

### 2.3.3 Cross-layer Activities

#### 2.3.3.1 Bandwidth Estimation

Due to similarities, the major techniques available to TCP for bandwidth estimation may also be used by SCTP, e.g., variable packet size (VPS) probing, packet pair/train dispersion (PPTD), self-loading periodic streams (SLoPS), and trains of packet pairs (TOPP) [44].

A combination of bandwidth estimates are used by Fracchia et al. to gauge the capacity of primary and secondary paths in [45]. A simple TOPP approach is used on the secondary path by sending a train of six variable sized packets (two small, two large, and two small) instead of the regularly scheduled heartbeats. At the receiver, the dispersion times of the large sized packets are measured and returned to the sender via a heartbeat acknowledgement. The bandwidth of an alternate path can be

calculated by simply dividing the size of either large packet by their corresponding separation time. Taking advantage of the continuous flow of data, bandwidth is estimated over the primary path from RTT and SACKs. A bandwidth sample is calculated from the $k^{\text{th}}$ RTT by

$$B_k = \frac{D_k}{\Delta_k}, \tag{2.4}$$

where $D_k$ is the reported number of bytes acknowledged by a SACK with a RTT of $\Delta_k$. A low pass filter is then applied to average the samples so the mean bandwidth is

$$\hat{B}_k = \alpha \hat{B}_{k-1} + (1 - \alpha) B_k, \tag{2.5}$$

where $\alpha$ is a constant, such that $\alpha \in (0, 1)$.

#### 2.3.3.2 Wireless Error Notification

Although TCP and SCTP are experts at handling common congestion along wired links, they are unable to distinguish errors caused by fickle wireless channels. If these undesired conditions are only temporary, e.g., from a sudden change in phase, congestion control reactions will be unnecessary and overkill. Already, a number of surveys have reported various solutions for handling TCP over wireless links [46–48]. Again, due to the close relationship between the two, the research efforts for TCP may also serve as excellent resources for SCTP's plight.

As an example of recycled research, the sender-side solution, TCP Westwood+, is used by SCTP to discriminate between congestion losses and wireless errors in [45]. The process works by comparing the output rate, i.e., CWND/RTT, with the most recent bandwidth estimate following a loss indication. If the output rate is larger than the bandwidth, congestion is inferred because all available bandwidth is being utilized. If this is not the case, however, a wireless loss is more likely, and instead of

dropping the CWND by half, it simply sets the slow-start threshold to

$$\text{SSTHRSH} = \frac{B \cdot \text{RTT}_{\text{min}}}{L_{\text{max}}}, \tag{2.6}$$

where $B$ is the bandwidth estimate, $L_{\text{max}}$ is the maximum packet size, and $\text{RTT}_{\text{min}}$ is the minimum observed round trip time. If four missing reports have triggered a loss indication, then the CWND is set to SSTHRSH. We should point out that bandwidth probing, required by the process, can obscure capacity measurements – something not considered by the researchers in their study.

Another approach uses the link layer at a wireless interface to fragment a bundled SCTP packet into its individual chunks. The ideology is that smaller packet sizes should have lower probability of transmission corruption over larger ones. Presented in [49], the process works by disassembling and reassembling bundled SCTP packets between a wireless receiver and its access point. Assuming all disassembled chunks are transmitted in-order, when the last chunk is received a SACK will inform the sender if any were lost. Senders noting that a lost chunk had been fragmented, will infer wireless errors as opposed to congestion loss. The effectiveness of this process, however, lies in the assumption that SCTP packets have more than one chunk. If only one chunk were sent per packet, there could be no inference. Furthermore, unless the receiver knows when it should have received the last chunk of a disassembled packet, it will not know how long to wait before reporting the packet missing.

Finally, TCP's explicit error notification (ECN) [50] mechanism can also be applied to SCTP. ECN is implemented in both the transport and network layers of the OSI model. At the network layer, routers determine congestion by comparing the current average queue length with some preconfigured threshold. When a router's queue length exceeds this threshold, it begins marking the ECN bit on all outgoing

IP packet headers. Transport layer receivers employing ECN can then inform corresponding senders of congestion by setting similar flags in TCP acknowledgements. In turn, senders may tune their output rates in a more informative manner. Intuitively, if loss indications are made without ECNs, then a wireless or noncongestive loss may be inferred. Since RFC 4960 has reserved two chunks for future ECN extensions, SCTP is poised to apply this well researched technique immediately. The work in [51] evaluates SCTP ECN and makes the following recommendations:

**(1)** Do not use the *congestion window reduced* (CWR) chunk as it consumes more bandwidth than its TCP counterpart through additional overhead.

**(2)** Only reduce the send rate once per window of congestion losses (i.e., treat remaining losses as noncongestive and perform normal retransmission procedures).

**(3)** To prevent underutilization, the minimum congestion window of an SCTP sender should be

$$\mathrm{CWND}_{\mathrm{min}} = \frac{B \cdot \mathrm{RTT}}{8} + \mathrm{MTU}, \tag{2.7}$$

where $B$ is the bandwidth of the path's bottleneck link and MTU is the maximum transmission unit.

#### 2.3.3.3 Network Intelligence

Assuming bandwidth information is available and unchanging, SCTP multihoming may be able to gain a performance advantage using asymmetric path selection. An asymmetric path is one where the forward and reverse paths do not have equal one-way delays. In fact, Internet users often subscribe to asymmetric paths when purchasing data plans from ISPs. Typically, a subscriber has higher download rates than

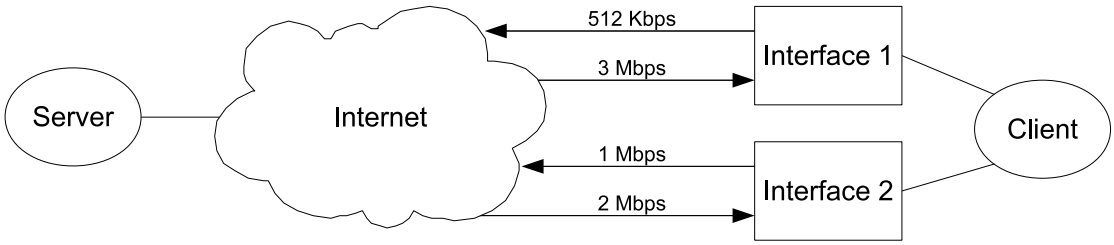


Figure 2.11: Asymmetric path selection.

they do for uploading. When an end-point is multihomed, the captured one-way delays could result in a significant improvement if used efficiently. Take for example the illustration in Fig. 2.11. Here we have a multihomed end-point with two interfaces. While interface 1 has download and upload rates of 3 Mbps and 512 Kbps, interface 2 offers rates of 2 Mbps and 1 Mbps, respectively. Clearly, better performance will be achieved using only one interface to download and the other to upload, than using a single interface for both. Recommended in [52], each sender should maintain a matrix of delays in each direction to generate all RTT combinations and optimize path selection. Referring to our simple example and assuming a constant packet size, to achieve the best performance, the server should transmit new data to interface 1, but send acknowledgements over interface 2.

Identifying shared bottlenecks is another strategy to make better use of system resources. Noted in [53], unique bandwidth estimates could be misleading if data streams share the same bottleneck link across separate network paths. Since the transport layer has a blind view of the network topology, the authors suggest comparing the autocorrelation in RTT delays of each path with the cross-correlation of combined paths. Then, two paths will share a common bottleneck if their cross-correlation is larger than the autocorrelation over one path. Upon this discovery, multiple transfers could be consolidated into one destination address – simplifying an SCTP association. Deallocating redundant interfaces, not only cures inefficiency, but

also frees resources for alternative uses.

## 2.4 Discussion

We now offer our own analysis of SCTP multihoming; commenting on the surveyed techniques and making recommendations for future research.

### 2.4.1 Handover Management

Preemptive path selection matches multihomed devices with the "best" available access network; but at what cost? Mobile devices are notorious for poor processing power, little memory, and limited battery life. Furthermore, applications like VoIP require rapid decision making results, something optimization cannot always provide. With respect to these caveats, optimal path selection is probably a long way off, though, existing approximation techniques can achieve good results in a reasonable amount of time, specifically, meta-heuristics. We suggest the preemptive path selection problem make use of trusted meta-heuristics like simulated annealing, tabu search, or genetic algorithms. Meta-heuristics, moreover, often converge in a fraction of time their optimal counterparts might take. Nevertheless, each application will need to define an objective function for the transport layer to solve. ECHO is a good framework for preemptive path selection because it defines an objective, i.e., maximize $R$. ECHO also translates link layer metrics (e.g., delay, jitter, transmission rate) into problem specific parameters. Because many network applications can be grouped together (e.g., voice, video, data), a path selection sublayer (between the application and transport layers) may prove beneficial to the future demands of the multihomed communication system.

Fault tolerance reduces the path selection problem to a reactive service. This should be considered a valid approach because the transport layer is already complicated enough, and the limitations of mobile devices will find optimal path selection taxing. Due to slow failure detection and unreasonable communication delays, however, SCTP's failover guidelines are unacceptable to most applications. Furthermore, the current research strategies place too much emphasis on sender-side failure detection; especially when channel information is at the receiver. Examining whether it is best to wait for the sender to initiate a primary path change after failover or have the receiver automatically reconfigure the connection based on its observed channel conditions, the data favours the latter. For instance, a number of articles have published results showing significant service disruption during SCTP based handovers [24, 54–56]. Using DAR, SCTP end-points can easily monitor their connection status and inform the sender of path changes. Unfortunately, if the receiving end-point waits too long, communication will be disrupted and even packets may still be lost if the sender is uninformed at the exact moment of path failure.

The fault tolerant path selection problem becomes more complicated when mobile devices move away from high bandwidth access points. Even if the signal strength of a new base station is stronger than its predecessor, the mobile will likely want to remain connected to the old base station as long as possible. It is our recommendation that handover prediction play a part in solving this problem. In most cases, handover prediction is used by base stations to make advance resource reservations for calls with high handover probability (i.e., calls most likely to transition from one cell to another). To make valid predictions, the mobile is usually fitted with a transceiver which feeds coordinates to a satellite positioning system. These coordinates are then sent to base stations in close proximity to the mobile, where electronic road maps are employed to make best guess approximations on handover time and location. The

wealth of research (e.g., [57–60]) in this area should pave the way for its assimilation with transport layer multihoming.

Moving the responsibility of path selection to the receiver should yield shorter interruption delays, but signalling will become more important if we want to maximize throughput. If a mobile finds itself crossing thresholds on a consistent basis, signalling and reconfiguration costs may only exacerbate the problem. A hysteresis can offer some form of stability, but should do so by adapting to environmental conditions. For example, in the same way RTO is tuned to the ever changing RTT, so should a hysteresis reflect the rate and trade-off of path switching.

Following handover, spurious retransmissions plague the entire communication system; while the network suffers from inefficient use, the end-point loses throughput momentum. So far, post handover synchronization is handled in one of two ways: (1) multipath transmissions (i.e., sending the same data over multiple paths), or (2) spurious retransmission detection. In either case, the system is treated through redundant measures. Similar to fault tolerant path selection, better prediction may also improve handover synchronization. By forecasting the change in connection quality (e.g., bandwidth, delay), SCTP should be more prepared and thus more capable of efficient handover management.

The techniques (from this paper) used for handover management are summarized in Table 2.1. While many approaches share the same characteristics, they differ in their implementation benefits.

### 2.4.2 Concurrent Multipath Transfer

A number of solid contributions has elevated the operational efficacy of CMT. By curbing the effects of reordering and stunted CWND growth, CMT can now achieve

Table 2.1: SCTP Handover Techniques

| Algorithm Name | Problem Addressed | Solution Approach | Bandwidth Aware | Benefits | Drawbacks |
|---|---|---|---|---|---|
| SIGMA | path selection | maximize single variable function | no | application independent | best-effort |
| ECHO | path selection | maximize multi-variable function | yes | QoS | application specific |
| SMART-FRX | losses, failover delay | multipath transmissions, retransmission policy | no | loss differentiation | failover scheme, redundant packets |
| MTA | losses | multipath transmissions | no | simple | failover scheme, redundant packets |
| Eifel | spurious retransmissions | detection | no | fast | unreliable |
| DSACK | spurious retransmissions | detection | no | reliable | slow |
| SHOP | reordering | smart scheduling | yes | receiver initiated | monitoring overhead |
| HR-mSCTP | reordering, losses | multipath transmissions | no | receiver initiated | redundant packets |

aggregate performance gains, but work on the scheduling front still remains. How to choose transmission paths to minimize receive buffer blocking is a challenging problem that needs to be addressed. The MDP approach can certainly aid heuristic design as long as transport layer constraints are kept in mind, but this may prove challenging.

Our own work focuses on the performance issues of CMT with limited RBUFs (see Chapter 4). Using intelligently scheduling, we can maintain a continuous flow of new data without interruption, while at the same time approaching the theoretical limits of aggregated performance. Our scheduling algorithm works as follows: after a CWND update, we estimate how many packets can be delivered over the faster path ahead of a single packet using the slower one; we then use this estimate to transmit the next packet (i.e., set of bytes) that will minimize buffering at the receiver. This is in contrast to the current scheduling policy, which simply sends the next sequential packet over each path regardless of performance disparity.

### 2.4.3 Cross-layer Activities

Without explicit feedback from Internet routers, SCTP end-points use RTT estimates to measure network performance capabilities. The various methodologies used by TCP can also be applied to SCTP. We should point out, however, the intrusive

nature of measurement techniques; that is, packets always need to be transmitted before any valid results can be attained. This makes the network susceptible to gratuitous degradation and inefficiency. Fortunately, some measurement strategies are less intrusive than others – though they cannot always provide the right amount of information. Since most of the mulithoming techniques we have discussed assume the availability of bandwidth conditions, care must be taken each time the network is measured.

Losses occurring over wireless hops imply a very different result than those experienced over a wire. Wireless losses usually occur at random times and are often caused by unforeseeable events like interference or sudden movement. Of the three approaches applied to SCTP, two show significant merit, that is, TCP Westwood+ and ECN. Though concern for both exist; while the former suffers from intrusive bandwidth probing, the latter has scalability issues. For instance, all routers must employ ECN, and there may be 10s or even 100s of routers between an SCTP sender and receiver. Without a defining solution, differentiating between wired and wireless losses should be a primary concern for multihomed devices as we embark further into the age of wireless computing.

Identifying shared bottlenecks and routing traffic at ingress points are just some of the newer responsibilities assumed by the transport layer with the assimilation of multihoming. Needless to say, with more research in this area, SCTP will likely evolve to take on even more roles than its transport layer predecessors ever imagined.

## 2.5 Summary

In this chapter, we have presented a comprehensive review of transport layer multihoming using the steam control tranmission protocol (SCTP). Currently, the main

areas of study include: handover management, concurrent multipath transfer, and cross-layer activities. A survey of strategies among research efforts has brought forth improved performance while in some cases even complete solutions. With that said, even though some algorithms may offer sufficient results, many open problems still remain.

# Chapter 3

# System Description

In this chapter, we first introduce the multihomed system by comparing its network topology and transport layer architecture to that of a more traditional (one interface) system. Next, we outline a framework for CMT by listing its required components. Finally, we summarize the mechanics of SCTP for use in later chapters.

## 3.1 System Comparison: Singlehomed vs. Multihomed

In this section, we introduce the multihomed system by comparing its network topology and transport layer architecture to that of a traditional system with only one interface.

### 3.1.1 Network Topology

Ordinarily, the transport layer is modelled as a dumbbell topology, that is, one bottleneck link dividing a set of senders from their respective receivers. More specifically, however, we typically study only one sender/receiver pair, while another $m$ make up background traffic sources. This model is true for singlehomed end-points only (i.e., one network interface). The dumbbell topology for a singlehomed connection is shown in Fig. 3.1. In the figure, $s_{\text{sctp}}$ and $r_{\text{sctp}}$ represent the SCTP sender and receiver;

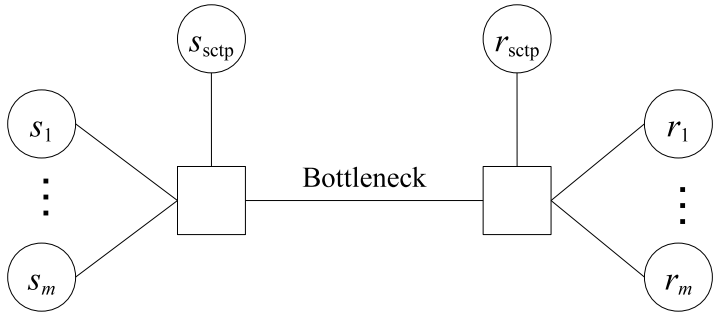


Figure 3.1: Singlehomed network topology.

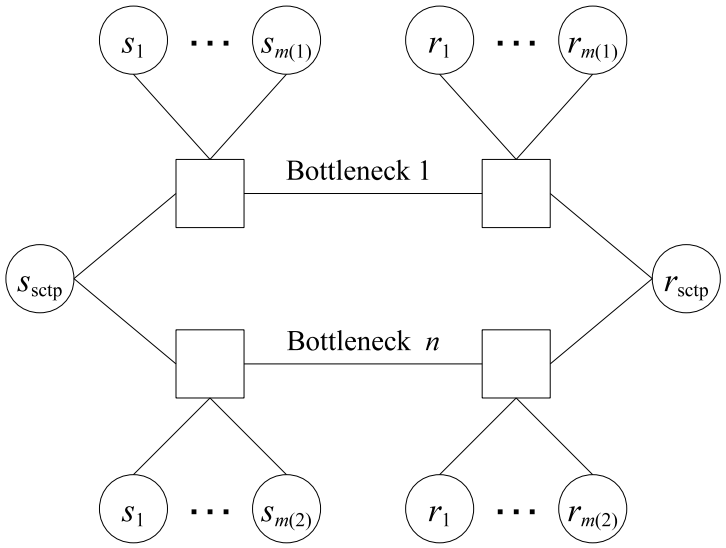


Figure 3.2: Multihomed network topology.

while $s_m$ and $r_m$ are respectively the $m^{\text{th}}$ background sender and receiver sharing the bottleneck link.

When an end-point is multihomed (i.e., more than one interface), the singlehomed dumbbell topology is no longer valid, since more than one network path exists between sender and receiver. For multihomed end-points, rather, the network is modelled as a series of dumbbells, where the function $m(n)$ returns the maximum number of background sources sharing the $n^{\text{th}}$ bottleneck link. Transmission and buffering capacity, moreover, is assumed to be independent of bottleneck link, so that loss rate

and end-to-end delay may differ from path to path. The network topology for a mulithomed system is given in Fig. 3.2.

### 3.1.2 Transport Layer Architecture

While the transport layer's primary responsibilities are to guarantee successful and ordered delivery of application data, a number of caveats will shape an end-point's design, such as limited buffer space, shared resources, and network uncertainty. Taking these caveats into consideration, a singlehomed end-point will implement the following: send buffer, receive buffer, retransmission timer, fast recovery/fast retransmit, flow control, and congestion control. The design of a singlehomed end-point is shown in Fig. 3.3.

**(1)** The send buffer (SBUF) accepts data from the application layer, converts it to packets, and stores it until transmission. Even after transmission, however, each packet remains in the SBUF until an acknowledgement confirms its successful, and ordered, delivery. Since the SBUF is finite in space, moreover, when it is full the application layer will be blocked.

**(2)** If the associated application is downloading data (i.e., the end-point is a receiver), the receive buffer (RBUF) will hold out-of-order packets until a CUMACK arrives, before passing them to the application layer. On the other hand, if the application is uploading data (i.e., the end-point is a sender), the RBUF will simply accept acknowledgements; notifying the sender of successful and/or out-of-order delivery.

**(3)** If packets are lost, deadlock is averted when the retransmission timer expires. Every time a packet is transmitted and the retransmission timer is not already

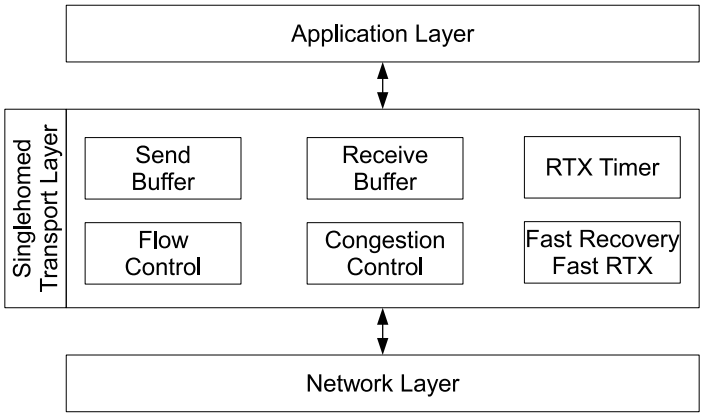


Figure 3.3: Singlehomed end-point architecture.

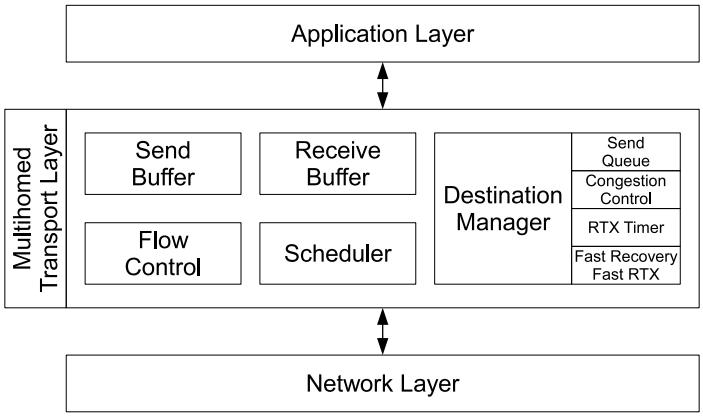


Figure 3.4: Multihomed end-point architecture.

running, it must be started. Then, if no acknowledgements are received within some duration of time, the sender will know a packet was lost. The mean and variance of sampled round trip times (RTTs) are used to determine a suitable duration for the retransmission timeout (RTO).

**(4)** Fast recovery/fast retransmit recognize loss before the retransmission timer expires by counting the number of times a packet is missing. A packet is considered missing if another packet, with a higher sequence number, is acknowledged first. If a packet reaches a missing count of four, then the packet is considered lost

and a retransmission is made. Losses triggered by fast recovery/fast retransmits also act as indicators of congestion; prompting the sender to slow down transmission and mitigate further loss.

**(5)** Flow and congestion control are used together to administrate transmission opportunity. Using a variable called the receive window (RWND), flow control[1] verifies the availability of buffer space at the receiver before allowing transmission. In another way, congestion control[2] employs the CWND to limit the number of packets that can be in flight (i.e., passed to the network layer but undelivered). Although the mechanics of flow control and congestion control might suggest redundancy, their mandates prove otherwise. While flow control circumvents losses at the receiver in the event of reordering or sluggish processing, congestion control aims to mitigate network loss from bottleneck overflow.

A mulithomed end-point shares many similarities with its singlehomed counterpart. For example, data is still copied from the application layer and transformed into packets before being placed in a buffer; as well as flow control continues to manage receiving capabilities. What has changed, rather, is how packet transmissions are coordinated among numerous destination addresses. The architecture of a multihomed end-point is illustrated in Fig. 3.4.

**(1)** The first change involves adding a scheduling module to the design. The scheduler provides fundamental service for CMT, but is especially crucial when performance characteristics (e.g., bandwidth, delay, loss rate) vary from path to

1. Flow control maintains that the receive window is greater than zero in order to allow a transmission.

2. Congestion control will only allow transmissions when the number of outstanding packets is less than CWND.

path. The scheduler's main job is to pair packets in the SBUF with destination addresses so that reordering at the receiver is minimized.

**(2)** The destination manager is the second major change for the multihomed endpoint. As we said before, performance characteristics will vary from path to path, therefore each destination address should maintain independent state variables (e.g., CWND, CUMACK, RTT), not only for scheduling purposes, but also to distinguish between reordering and loss. For example, under the singlehomed network topology, packets are expected to arrive in-order, since they should (for the most part) traverse the same first in first out (FIFO) network path; it would then be reasonable to assume loss after a small number of packets have been reported out-of-order. The multihomed network, however, will reorder packet arrivals on a more frequent basis. Therefore, if we want to use mechanisms like fast recovery/fast retransmit, we need to manage them separately for each destination address. Since the retransmission timer and congestion control mechanisms also use path specific variables, they too are now controlled by the destination manager.

**(3)** Finally, each destination address is associated with its own send queue. This space does not need to be allocated explicitly, but can be virtually extrapolated from the SBUF. Typically, a destination's send queue is used to determine if a packet has been scheduled (i.e., assigned to a destination), and/or transmitted to a particular destination.

## 3.2 Concurrent Multipath Transfer

Primarily, CMT is a framework for sending data to a receiver with multiple IP addresses. Although there is no mandate on service type, CMT is typically thought of as a reliable service. Since SCTP satisfies both preconditions, CMT is often an extension of SCTP[3]. To support CMT, SCTP needs to implement the following: 1) multihomed congestion control, 2) packet scheduling, and 3) congestion window management.

### 3.2.1 Multihomed Congestion Control

If the paths between sender and receiver are assumed independent, then network characteristics (e.g., bandwidth, delay, loss rate) will vary between addresses. Traditional congestion control senses network capacity by slowly increasing the number of packets in flight (i.e., unacknowledged). When packets are lost, this number is decreased by half to reflect the presence of congestion. But it is how the sender notices loss that makes congestion control a problem for CMT. Every time a packet arrives at the receiver a SACK tells the sender if any are out-of-order. If only one network path separates sender from receiver and packets were transmitted in sequential order, it is intuitive to infer loss when packet arrivals are persistently out-of-order.

When multiple network paths are available (such is the case with CMT), packets are routinely received out-of-order because of disparate path characteristics. This results in spurious loss events slowing transmission – unnecessarily – even when no congestion exists. One solution is to implement congestion control separately for each IP address; for example, each address maintains its own CWND and CUMACK. The work in [30] outlines the algorithms necessary to implement multihomed congestion control for CMT.

3. TCP is unsuitable for CMT since it does not support multiple IP addresses.

### 3.2.2 Packet Scheduling

Maintaining reliability alone is hardly a challenge; if data is out-of-order, the sender will be notified by a SACK and retransmissions will correct the problem. This is the typical situation when packets are lost and is a reasonable solution. CMT creates a new problem, however, even in the absence of loss; when network paths are disparate, packets routinely arrive out-of-order. To make matters worse, if the RBUF is limited in size, new transmissions over faster paths are disrupted while out-of-order packets are buffered at the receiver; ultimately, reducing throughput to the capacity of the slowest path. A packet scheduler (see Chapter 4) can mitigate these effects by intelligently assigning outgoing packets to the destination address that minimizes buffering delay at the receiver.

### 3.2.3 Congestion Window Management

Even with multihomed congestion control and ordered packet arrivals, inefficiency can still plague CMT. The source of weakness actually stems from flow control; a common mechanism found in TCP and SCTP. Flow control is designed to prevent a sender from overwhelming the receiver; for example, if the sender transmits packets faster than the receiver can process them, the system becomes unstable and creates unnecessary loss. To circumvent this problem, flow control prevents the number of packets in flight (including those that are buffered at the receiver) from exceeding the size of the RBUF. Another way of solving this problem is to simply limit the CWND to the size of the RBUF.

In the case of CMT, the same method can be employed, but problems occur when the bandwidth delay product (BDP) of one path is disproportionate compared to others. Ultimately, the BDP can tell us how many packets can fit onto a path at

any given time; or in other words, a limit to place on the CWND. On the contrary, if CWNDs remain unbounded and the path to one destination address has an exceptionally large BDP, utilization can drop leading to an inefficient use of resources. Furthermore, if the sum of BDPs (for each path) exceeds the size of the RBUF, CWND size becomes the determining factor for system optimization. For example, if we can only afford to increase the CWND of one destination address, we should increase the one with the lowest RTT (or the highest bandwidth) in order to maximize throughput. Congestion window management will be presented in Chapter 6.

## 3.3 Stream Control Transmission Protocol

We now make the following assumptions regarding the stream control transmission protocol (SCTP) [2]. Note that these assumptions will carry forward through the remainder of the thesis.

**(1)** The amount of DATA the sender is allowed to transmit to address $a$ at time $t$ is controlled by the address's CWND, its number of outstanding bytes (OUT) (i.e., any unacknowledged data), and the RWND. At any given time, this amount is the lesser of the CWND minus OUT and RWND. Thus

$$\text{DATA}_{a,t} = \min(\text{CWND}_{a,t} - \text{OUT}_{a,t}, \text{RWND}_t). \tag{3.1}$$

**(2)** Depending on mode, SCTP increases a CWND in two ways:

**(*a*)** An address is in slow-start (SS) mode if its CWND is less than the SSTHRSH. Every time a CUMACK is received, the CWND is increased by at most, the lesser of *i.* the number of outstanding bytes being newly acknowledged

(NEWACK) (i.e., the number of packets removed from the send buffer), and *ii.* the maximum transmission unit (MTU). The new CWND is thus

$$\text{CWND}_{a,t+1} = \min(\text{NEWACK}_{a,t}, \text{MTU}). \tag{3.2}$$

**(*b*)** An address is in congestion avoidance (CA) mode if its CWND is greater than or equal to its SSTHRSH. In this mode the CWND is incremented by one MTU every time a sender receives an acknowledgement that advances the CUMACK, and the CWND is less than or equal to the partially acknowledged bytes (PBA) and OUT. This is accomplished by reducing the number of PBA by the previous size of the CWND (i.e., the size of the CWND before the increase). The new CWND is calculated by

$$\text{CWND}_{a,t+1} = \text{CWND}_{a,t} + \text{MTU}. \tag{3.3}$$

**(3)** Every time the sender receives a SACK, NEWACK is subtracted from OUT. In other words, new data can be sent without increasing the size of the CWND (i.e., the CWND is opened).

**(4)** When gaps are found in a SACK, the sender increments the missing count for any packets contained within a gap. If the SACK indicates that a packet has been missing four times, the packet is considered lost. In response, the sender halves the CWND of the address to where the packet was sent, then retransmits any lost packets. Moreover, the address's SSTHRSH is also set to the size of the halved CWND, in case of a timeout event.

**(5)** Each time new data is sent to an address, a timer called the retransmission

timeout (RTO) is set. If this timer expires before an acknowledgement arrives for the transmitted data, the CWND is dropped to one and RTO is doubled. This process continues for every consecutive packet loss until some maximum RTO is reached.

**(6)** RTO is calculated from two variables, the smoothed RTT (SRTT) and the average variation in RTT (VRTT). Every time there is a CWND update, new values for SRTT and VRTT are computed using exponential weighted moving averages. While the variation in RTT is given by

$$\text{VRTT} = (1 - \beta)\text{VRTT} + \beta|\text{SRTT} - \text{RTT}|, \tag{3.4}$$

the smoothed RTT will be

$$\text{SRTT} = (1 - \alpha)\text{SRTT} + \alpha\text{RTT}, \tag{3.5}$$

where $\alpha = \frac{1}{8}$ , and $\beta = \frac{1}{4}$.

## 3.4 Summary

In this chapter, we described how multihoming affects a traditional (one interface) system. Specifically, we showed how network topology and transport layer architecture change when end-points are multihomed. Furthermore, our basic assumptions, with regards to SCTP, were presented in detail. With a basis for our system of study now formalized, the more technical contributions of the thesis will be presented in the chapters to follow.

# Chapter 4

# CMT: Scheduling

Aggregating the resources of multiple network paths for a single transport layer session can improve throughput performance [30–32]. Unfortunately, improvement is restricted to the assumption of a infinite RBUF; which is not always realistic, especially for mobile devices with limited memory. When the RBUF is limited, aggregated performance diminishes; primarily due to *naive* round robin scheduling. Additionally, when multiple paths have disparate performance characteristics, such as round trip times, throughput can reduce to the capacity of the slowest path [34]. The problem exists, mainly, from a need to transfer data reliably and in sequential order to a destination with limited room for buffering. The research community has dubbed this the *receive buffer blocking* problem.

In this chapter we propose the on-demand scheduler (ODS) to address receive buffer blocking under delay-based disparity. ODS differs from previous scheduling algorithms in that it waits until congestion and flow control allow transmission before assigning a packet to a destination address. ODS, moreover, is particularly useful when the RBUF is small and the sender is unsure whether CMT is better than using only one network path.

The rest of the chapter is organized as follows: the current scheduling algorithms for CMT are described in Section 4.1; ODS is presented in Section 4.2; Section 4.3 provides performance results; and lastly, Section 4.4 summarizes the work.

## 4.1 Current Scheduling Algorithms for CMT

The purpose of this section is to describe, in detail, the mechanics of current scheduling algorithms for CMT. Note that a review of techniques aimed at naive scheduling as well as the receive buffer blocking problem were presented in Section 2.3.2.

### 4.1.1 Naive Round Robin Scheduling

The naive scheduler needs minimal explanation as it is simple and straight forward. Whenever transmission to destination $d$ is granted (i.e., RWND $> 0$ and $d$'s CWND is greater than the number of packets outstanding to $d$), the cumulative packet from the SBUF is assigned to $d$ and then passed to the network layer for transmission. Alternatively, when multiple destination addresses have transmission opportunities, the scheduler iterates through a list in a round robin fashion; each time assigning the cumulative packet to another destination address. Using this method, no priority is given during the scheduling process, making its implementation trivial. An unfortunate consequence of naive scheduling, however, is the fact that aggregated performance reduces as path characteristics become increasingly disparate. In the end, no destination, regardless of delay, can deliver packets faster than the speed of the slowest path.

### 4.1.2 Bandwidth Aware Scheduler

Since naive scheduling uses no intelligence, reordering problems can occur when path characteristics (e.g., bandwidth or delay) are disparate in nature. As a consequence, the bandwidth aware scheduler (BAS) was designed to remedy a solution. By assigning newly created packets to the "best" destination address before transmission,

BAS attempts to mitigate the effects of reordering caused from disparate path characteristics. When a new packet arrives in the SBUF, BAS assigns it to a destination address and places it in a virtual send queue. Later, when a destination has a transmission opportunity, the packet with the lowest sequence number is retrieved from the destination's virtual send queue and passed to the network layer.

BAS makes scheduling decisions based on bandwidth estimates as well as the volume of undelivered packets for each destination address. Each time a packet arrives in the SBUF, BAS computes the *reception index* of a destination address. The reception index combines a destination's load (i.e., number of scheduled packets) with its capacity (i.e., rate of delivery) so a sender can make quick, but informed scheduling decisions. A new packet $i$ is always assigned to the destination with the lowest reception index. Reception index, $R(d)$, is calculated by

$$R(d) = \frac{L(i) + O(d) + S(d)}{B(d)}, \tag{4.1}$$

where $S(d)$ represents the current number of bytes scheduled for destination $d$, $O(d)$ returns the number of bytes in flight (i.e., how much has been transmitted to $d$, but yet to be acknowledged), $B(d)$ is the bandwidth estimate for destination $d$, and $L(i)$ returns the size of packet $i$.

When transmission to a destination is allowed, a scheduled packet is simply retrieved from a queue and passed to the network layer. What happens, however, when transmission to multiple destinations is allowed at the same time? In this case, the sender needs to decide which destination it should send to, or in other words, which queue to send from. Unfortunately, the answer is nontrivial and BAS does not offer any advice on the matter. Since no solution is provided, we propose a simple round robin transmission policy. For instance, when transmission to multiple

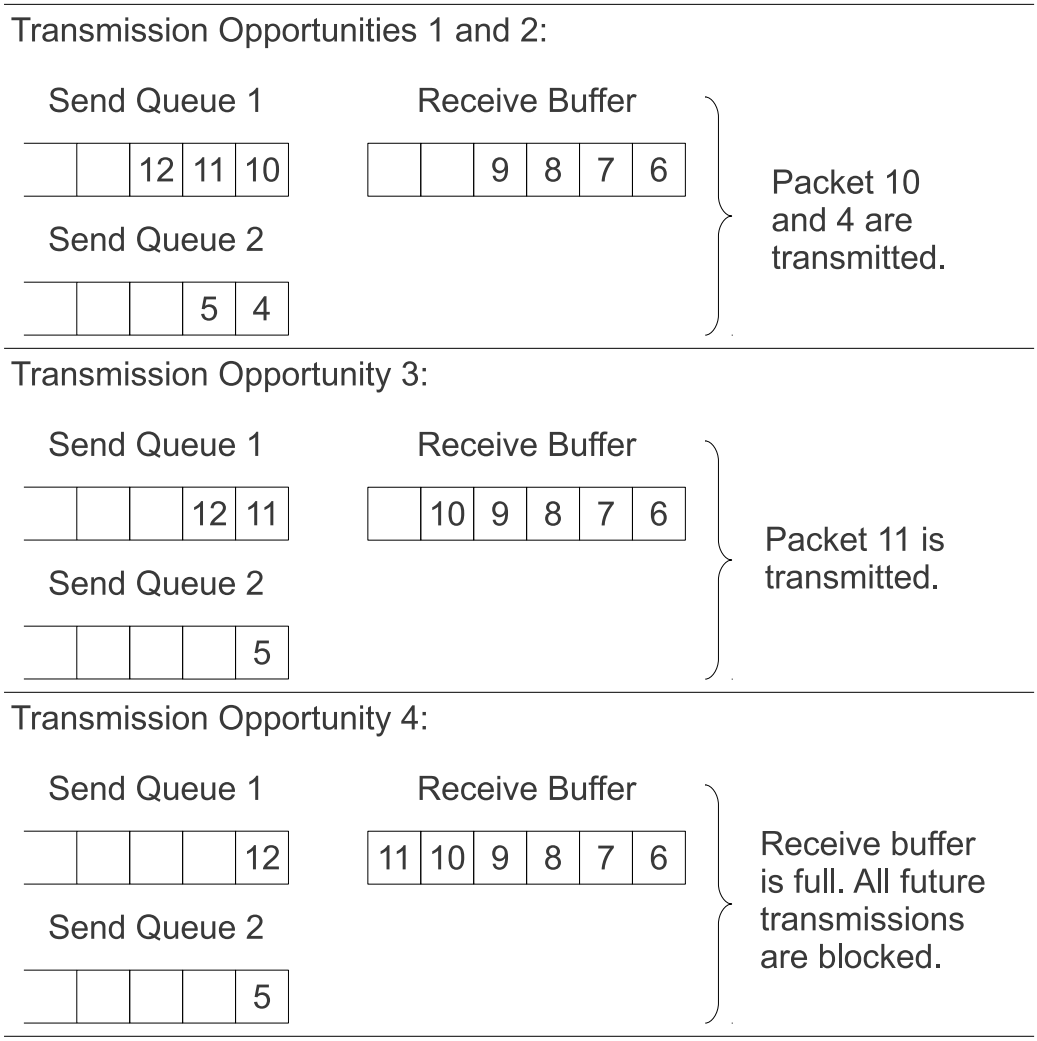


Figure 4.1: Round robin transmission policy failure.

destinations is permitted, the sender iterates between virtual send queues, allowing one transmission each. We will now show how even this simple scheme can fail.

Assume a dualhomed receiver (i.e., two destinations) is capable of holding up to six packets in its RBUF. Furthermore, let us assume 12 packets have been scheduled so far, where 1 through 3 and 6 to 9 have been delivered successfully, but packets 4 and 5 were lost. If packets 4 and 5 were scheduled for destination 2 while 10 through 12 have been scheduled for destination 1, it is possible for the association to become deadlocked depending on which queue transmits first. Referring to Fig. 4.1, we will

illustrate how this scenario can play out. Starting from the top, if the sender is employing a round robin scheme and packets scheduled to destination 1 are next in line for transmission, packet 10 followed by packet 4 are sent. Upon their arrival at the receiver, packet 4 is immediately passed to the application layer (since it is cumulative) but 10 is buffered (along with 6 through 9) since 5 is still outstanding. At the beginning of the next transmission opportunity, queue 1 will again be next in line, so packet 11 is sent. Unfortunately, this will fill the RBUF to capacity and no more transmissions will be allowed unless the receiver drops at least one packet, that is to say, it reneges.

One solution simply mandates that cumulative packets must be transmitted when the RWND is one packet long. In our scenario, however, the association may find it more desirable to send packets 4 and 5 immediately (i.e., during the first transmission opportunity), or for that matter, always choose the queue with the lowest sequence numbers, even before RWND = 1. If more packets have been scheduled for transmission, always sending from the queue with the lowest sequence number might move data out of the RBUF faster; possibly creating more transmission opportunities and increasing throughput. On the other hand, choosing queues based on sequence number may undermine BAS, creating redundancy in the system. Since our work is geared toward a different scheduling approach, i.e., one that assigns packets only after transmission is granted, we leave BAS scheduling problems to the research community.

## 4.2 On-demand Scheduler

Using BAS, scheduling is performed before a transmission opportunity is even available. For ODS, however, the opposite is true; that is, the destination is selected first

and a packet is chosen second.

Every time the sender has a transmission opportunity, ODS ranks its set of destination addresses using an estimated time of acknowledgement (ETA)[1] for cumulative packet $i$. The ETA of packet $i$ transmitted to destination $d$ is then calculated by

$$A(i, d, t) = t + K(d) + G(d, t), \tag{4.2}$$

where $t$ is the time of transmission and $K(d)$ returns the average RTT to destination $d$. The function $G(d, t)$, moreover, provides an estimate on queueing delay at $d$'s bottleneck link. The queueing delay associated with destination $d$ at time $t$ is

$$G(d, t) = \begin{cases} H(d) + T(d) - t & \text{if } H(d) + T(d) > t, \\ 0 & \text{otherwise,} \end{cases} \tag{4.3}$$

where $T(d)$ returns the transmission time of the last packet sent to destination $d$, and $H(d)$ estimates the length of time before the last packet transmitted to destination $d$ exits the bottleneck link. $H(d)$ is then updated after every transmission so that

$$H(d) = G(d, t) + \frac{L(i)}{E(d)}, \tag{4.4}$$

where $E(d)$ is the bandwidth of destination $d$'s bottleneck link[2].

First, ODS must discover the destination with the earliest ETA, that also has an open CWND (i.e., allowed to transmit); we will refer to this destination as $d_{\min}^{\text{Tx}}$. If $D$ represents the set of all destination addresses, then $d_{\min}$ will be the destination

1. ODS considers ETAs to be the time packet $i$ is acknowledged at the sender.

2. A path's bandwidth can be found using the dispersion in arrival times between two packets that were transmitted back-to-back.

with the earliest ETA. If $d_{\min} = d_{\min}^{\text{Tx}}$, then cumulative packet $i$ is sent to $d_{\min}$. On the other hand, if $d_{\min} \neq d_{\min}^{\text{Tx}}$, then $d_{\min}$ is removed from $D$, placed into $D'$, and the destination with the next earliest ETA is selected. Finally, when $d_{\min}^{\text{Tx}}$ is found, an algorithm searches the SBUF for a packet no destination from $D'$ can deliver sooner.

Second, ODS searches the SBUF, looking for the packet with the lowest sequence number; such that sending to $d_{\min}^{\text{Tx}}$ now, will avoid reordering later. ODS accomplishes this by assuming all outstanding packets are successful so that the time of future transmission opportunities can be approximated. More specifically, acknowledgements are simulated using the ETAs of outstanding packets. After a packet is acknowledged, a new transmission might be allowed. When a transmission opportunity exists, another $d_{\min}$ and $d_{\min}^{\text{Tx}}$ are found using $D'$. At this point, however, if again $d_{\min} \neq d_{\min}^{\text{Tx}}$, then a new search is spawned from the current process; much like a recursive function. For example, assume a receiver has three destinations: $d_1$, $d_2$, and $d_3$. If the scheduler ranks the destinations according to ETA, so that $d_1 < d_2 < d_3$, when looking for a packet to send to $d_3$, a new search might be needed for $d_2$. When the search for $d_2$ concludes, the search for $d_3$ resumes. The pseudo code for ODS is presented in Appendix A.

## 4.3 Performance Evaluation

In this section we compare the performance capabilities of each scheduling algorithm under various network conditions when CMT is employed during a 1 GB file transfer. To provide unbiased results, we developed a simplified SCTP module using the *ns*-2 software package. Our SCTP module is capable of supporting each of the scheduling algorithms previously described in the chapter (i.e., naive, BAS, and ODS), as well as using only one path at a time.

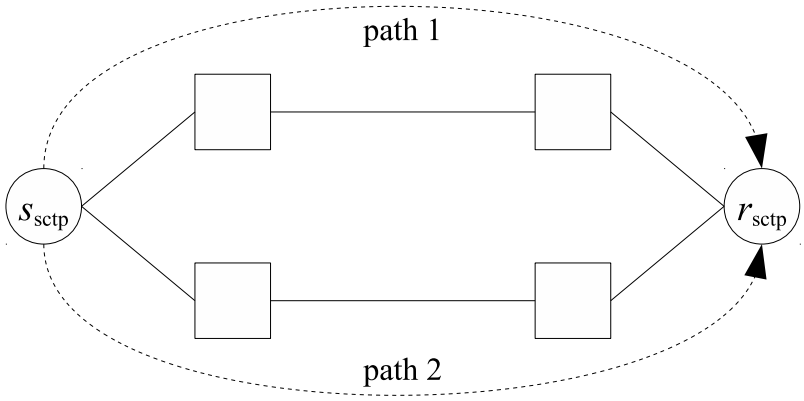


Figure 4.2: Network topology.

### 4.3.1 Evaluation Procedure and Network Topology

Fig. 4.2 illustrates the network topology used in our study. A two-path network was chosen because we expect CMT to be used by smartphone devices like the iPhone or BlackBerry Bold. At present, these devices come standard with two network interfaces, one for Wi-Fi and another for cellular connectivity. Assuming we are limited by these architectural constraints, only two network paths can exist between end-point devices. On the other hand, our results can only offer trends since many liberties were taken in an effort to reduce simulation complexity. For example, a network path in our system only consists of one link; where a link is composed of a transmission rate, queue size, and propagation delay. Furthermore, we always assume the return path to be infinitely faster than its forward counterpart. Finally, to create path disparity, specific parameters, e.g., bandwidth, delay, and loss rate, were varied along one of the paths. Unless otherwise specified, Table 4.1 lists all other simulation parameters.

| | | |
|---|---|---|
| application layer | file size | 1024 MB |
| SCTP end-points | packet size | 1500 bytes |
| | SBUF | 512 KB |
| | RBUF | 64, 128, 192, and 256 KB |
| | $RTO_{max}$ | 60 seconds |
| | RTO parameters | $\alpha = 0.125, \beta = 0.25$ |
| network paths | bandwidth | 10 - 50 Mbps |
| | propagation delay | 10 - 100 ms |
| | prob. packet loss | $10^{-4}$ - $10^{-1}$ |

Table 4.1: Comparison of scheduling techniques: simulation parameters.

### 4.3.2 Delay-based Disparity

In our first set of experiments, we evaluated the different scheduling algorithms under delay-based disparity (i.e., packet delay on one path is different from another). This was accomplished by varying the propagation delay on path 1 between 10 to 100 ms while the delay on path 2 remained constant at 40 ms. Other path characteristics, like bandwidth and loss rate, stayed the same on either path (i.e, bandwidth = 21 Mbps and loss rate = $10^{-4}$).

The results of our comparison are shown in Figs. 4.3 - 4.6. In all plots, throughput diminishes as delay increases. This happens because delay increases bandwidth delay product (BDP)[3]. To maximize throughput, the amount of data in flight (at any given time) must be the same as the BDP of the corresponding network path. Unfortunately, if the RBUF is lower than the BDP, then maximum throughput can never be achieved.

3. BDP tells us how much data can exist along a network path at any given time.

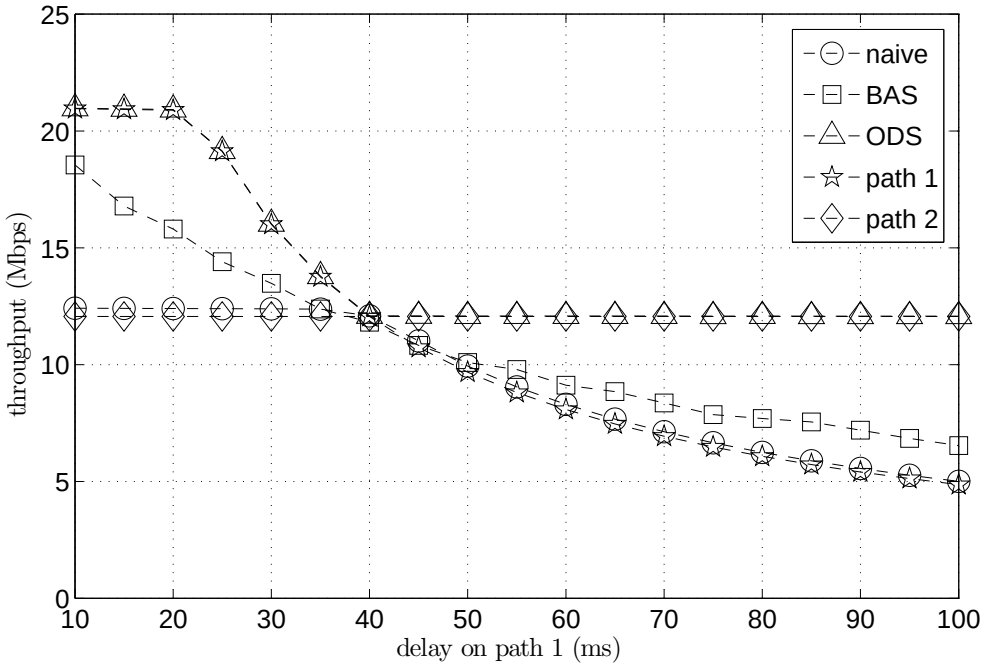


Figure 4.3: Delay-based disparity: throughput results when RBUF is 64 KB.

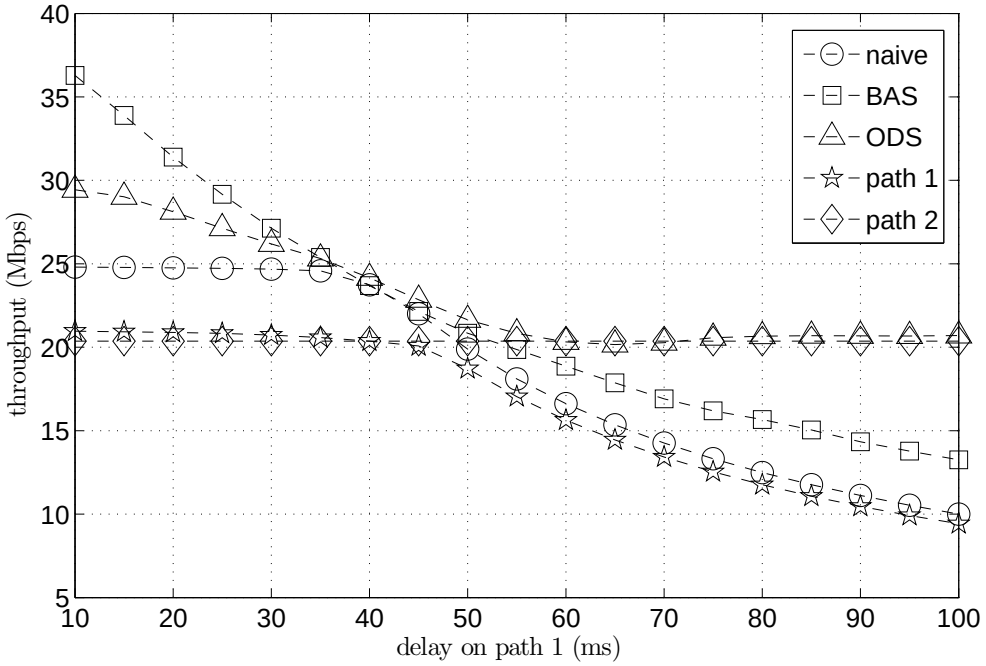


Figure 4.4: Delay-based disparity: throughput results when RBUF is 128 KB.

At a lower RBUF size, however, ODS can still offer improvement over the other two scheduling approaches, even when delay is at its highest (see Fig. 4.3). Although ODS cannot improve throughput beyond the capabilities of using only one network path, ODS takes the *guess work* out of the problem. For instance, when delay on path 1 is less than 40 ms, path 1 is clearly the better choice; but once delay goes beyond 40 ms, the sender should switch to path 2. Unfortunately, unless throughput prediction is available, choosing the better path can be a difficult decision to make. Furthermore, using only one path when the RBUF is small (i.e., 64 KB) always performs better than BAS or the naive approach. Alternatively, ODS is never worse and no decision needs to be made.

Similar results are observed in Fig. 4.4, with only one exception, BAS is best when delay on path 1 is low (i.e., less than 30 ms). In this case, we believe that SCTP's CWND update policy is the reason for ODS's poor performance. Since SCTP's CWND update policy grows the size of a CWND every RTT, the size of one CWND can be much larger than another if RTTs are different between paths. Moreover, even when flow control prevents transmission, a path's CWND can grow without bound. If loss rate is low (as was the case in our experiments), then ODS can allow one path to monopolize shared resources (e.g., the RBUF), due to an increase in transmission opportunities. Therefore, making scheduling decisions only after a transmission opportunity has been granted can actually decrease performance under certain conditions. SCTP's CWND update policy is less of an issue for BAS because scheduling decisions are made before transmission opportunities.

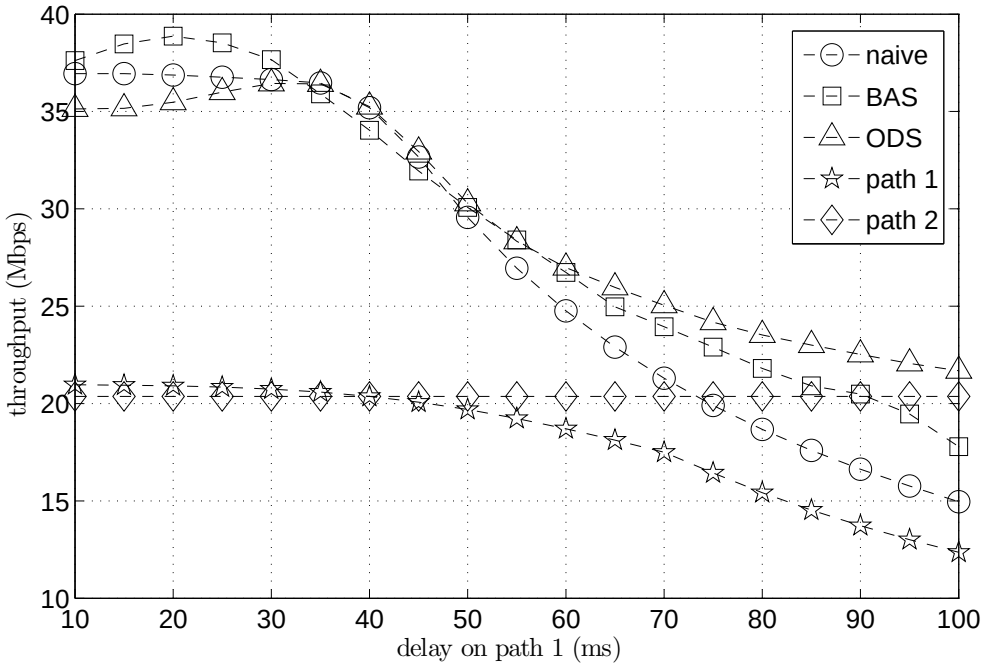


Figure 4.5: Delay-based disparity: throughput results when RBUF is 192 KB.

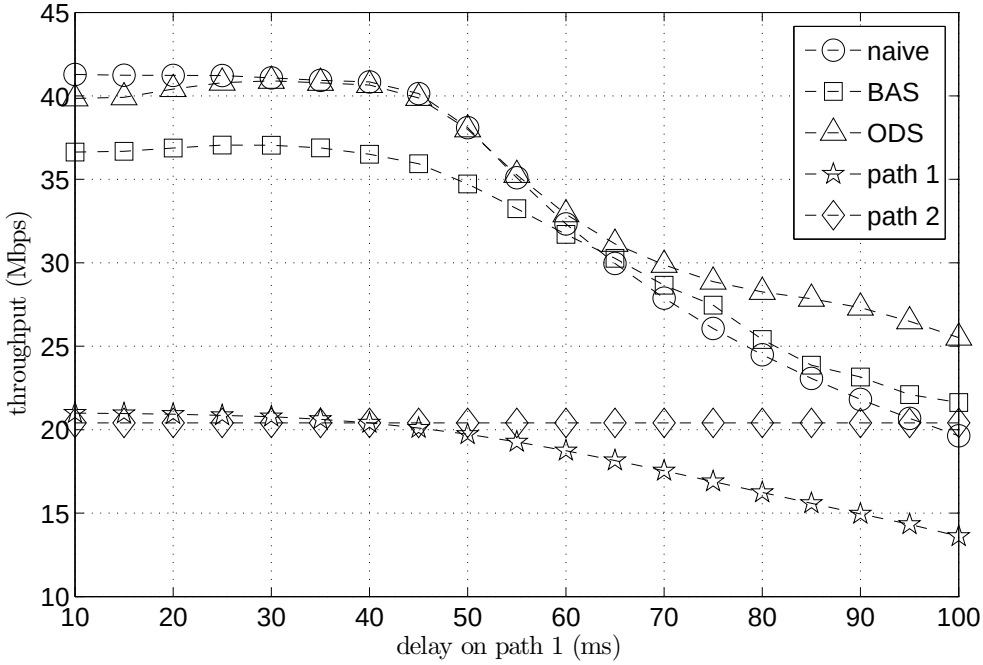


Figure 4.6: Delay-based disparity: throughput results when RBUF is 256 KB.

### 4.3.3 Bandwidth-based Disparity

We now evaluate the scheduling algorithms under bandwidth-based disparity (i.e., when available bandwidth on one path is different from another). Throughput results are shown in Figs. 4.7 - 4.10. Here, the naive approach is just as good as ODS when the RBUF is small; the reason stemming from marginal differences in delay. In these experiments, we kept the delay of each path to just 40 ms, but varied the bandwidth on path 1 from 10 to 50 Mbps while keeping path 2 at 21 Mbps. Although additional bandwidth (i.e., 50 Mbps) will decrease end-to-end delay and create some delay-based disparity, the difference is negligible compared to a 40 ms propagation delay. Ultimately, during bandwidth-based disparity out-of-order packets, caused by naive scheduling, wait only a small amount of time before receiving a CUMACK.

Interestingly enough, BAS is very poor when there is large disparity in bandwidth potential. Our understanding is that the naive scheduler creates reordering because of round robin transmissions, but every transmission sends a cumulative packet. On the other hand, BAS schedules packets prior to transmission but places them into a virtual send queue with the expectation that ordered packets will leave the queue at the same rate as its corresponding bandwidth estimate. Compared to naive scheduling, reordering caused by BAS is more consequential, as cumulative packets are moved from the RBUF to the application layer in a less frequent manner. Therefore, BAS needs a larger RBUF to mitigate the effects of these "scrambled" packet arrivals during bandwidth-based disparity.

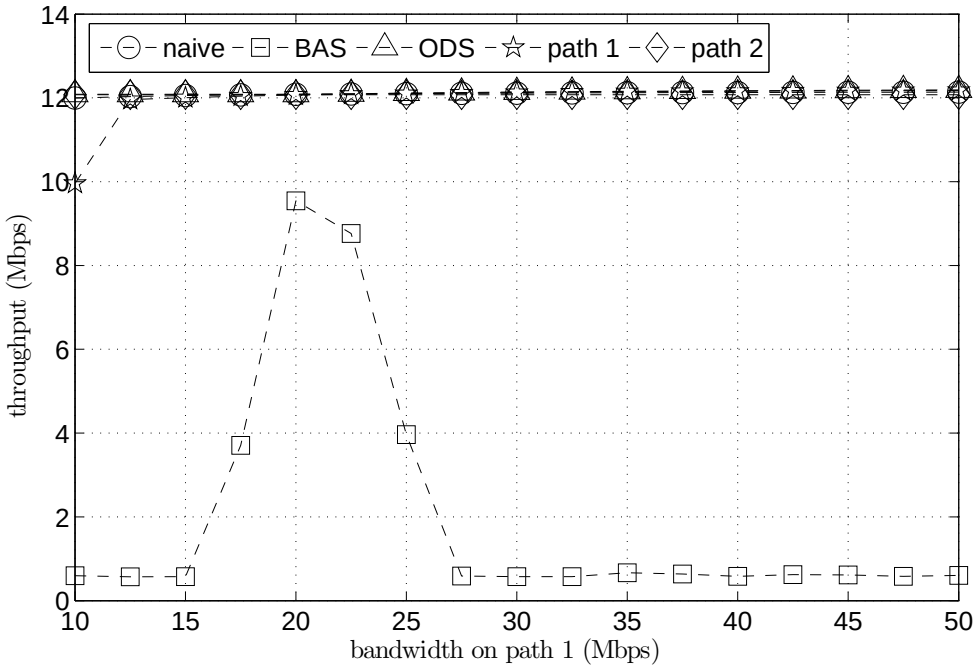


Figure 4.7: Bandwidth-based disparity: throughput results when RBUF is 64 KB.

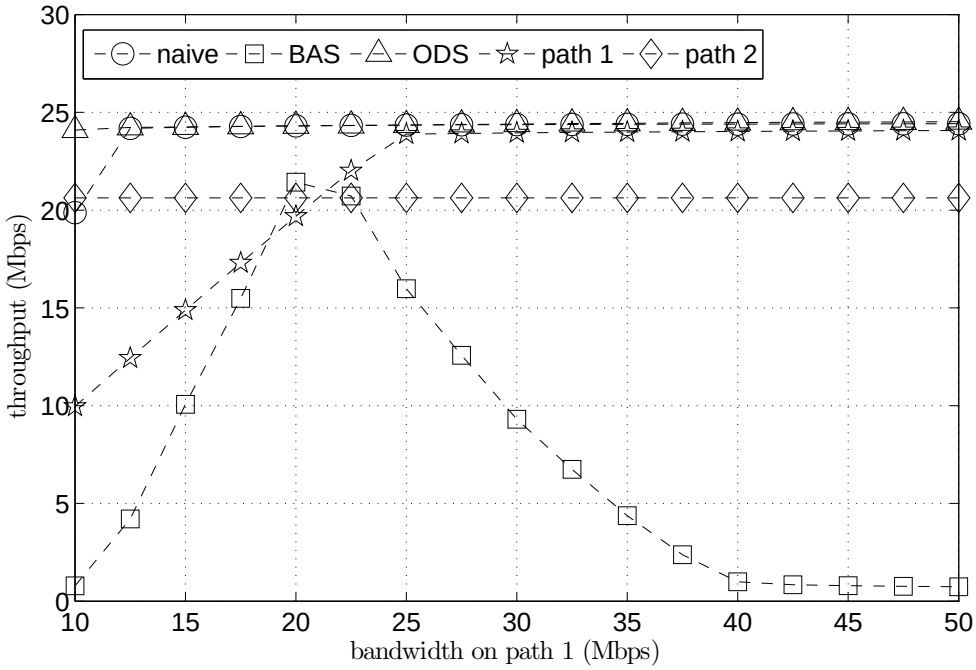


Figure 4.8: Bandwidth-based disparity: throughput results when RBUF is 128 KB.

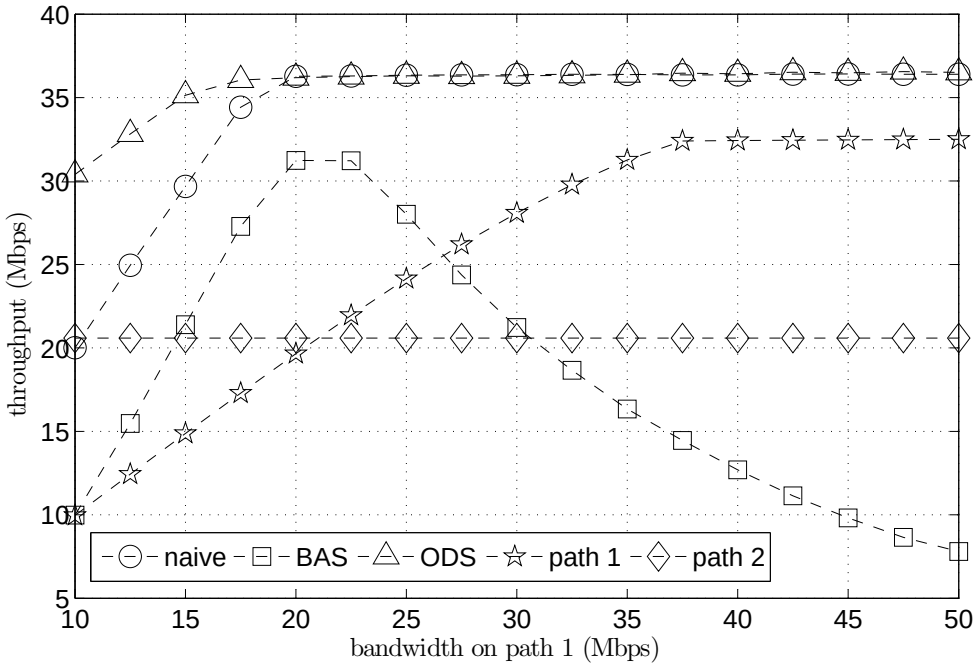


Figure 4.9: Bandwidth-based disparity: throughput results when RBUF is 192 KB.

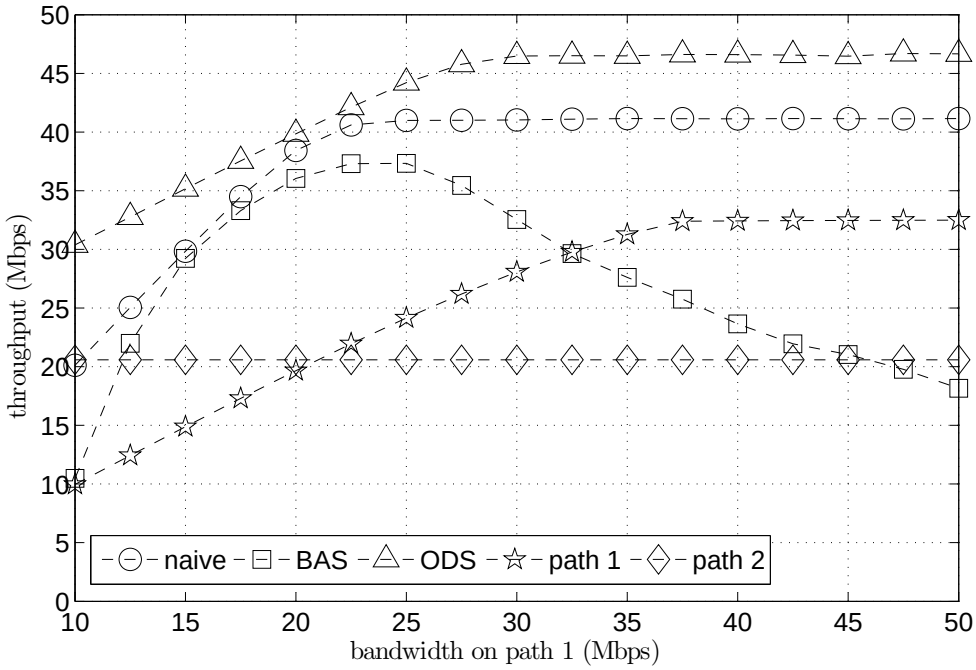


Figure 4.10: Bandwidth-based disparity: throughput results when RBUF is 256 KB.

The naive scheduler, however, becomes less effective with a larger RBUF. Even though bandwidth aggregation is still achieved, the faster path (i.e., path 1) cannot exceed the speed of the slower path (i.e. path 2). Of course, this is a consequence of naive scheduling, and results in receive buffer blocking slowing down the faster path. At any rate, ODS is not limited by this fact and can outperform the naive scheduler at higher RBUF sizes.

### 4.3.4 Loss-based Disparity

Next, we will look at how the scheduling algorithms fare under loss conditions. By varying probability of packet loss on one path, but keeping other characteristics the same (i.e., bandwidth = 21 Mbps and delay = 40 ms on both paths), we measure throughput using four different RBUF sizes; results of which are shown in Figs. 4.11 - 4.14.

Like the bandwidth tests of the previous subsection, ODS is no better than the naive scheduler. In fact, the naive scheduler performs slightly better than ODS, regardless of RBUF size. What these experiments tell us is that the receive buffer blocking problem is not immune to losses. Strictly speaking, ODS was designed to handle delay-based disparity by coordinating packet transmissions for ordered arrivals. In the case of loss-based disparity, however, ODS makes no adjustment for the possibility of packet losses. Therefore, receive buffer blocking will still exist when retransmissions are needed to reorder packets at the receiver. When a packet is lost, those that were scheduled to arrive after the loss will have to wait in the RBUF for a retransmission. Theoretically speaking, if loss rate is consistent, it may be possible to predict loss events. Knowing that a loss may occur, a second (more reliable) path could transmit redundant packets in an effort to save retransmission time. Such a

scheduler should be considered in future work.

Similarly, BAS is also unaware of loss probability. But BAS fairs much worse because packets are never removed from a send queue; even after loss, the BAS scheduler will continue to transmit along the same destination path. While a retransmission policy may mitigate some of these effects, placing packets into a send queue preemptively, with the expectation of successful delivery, will not overcome receive buffer blocking from loss-based disparity.

### 4.3.5 Destination Utilization

Even though ODS has been shown to be an effective scheduling algorithm for CMT in most situations, there are still some areas where this is untrue. For example, in Fig. 4.4, BAS performed better than ODS when delay on path 1 was less than 30 ms. In this subsection, we will examine efficiency in terms of utilization. Here, we define utilization as the number of transmissions to a destination divided by the number of transmission opportunities. For example, when a destination's CWND is open (i.e., the number of outstanding packets is less than the size of the CWND), a destination will have a transmission opportunity. Unfortunately, if the RWND is less than or equal to zero, then transmission is blocked.

In Figs. 4.15 and 4.16 we plot the utilization and throughput on path 1 and 2 when ODS is employed[4]. As we can see in the graphs, both utilization and throughput are higher on path 1 than on path 2, but this is be expected since path 1 has a lower delay. Moreover, the RBUF is limited; therefore, path 2 should yield to path 1 whenever the RWND is zero. Even so, utilization is remarkably low on path 2, below 10% even when delays on either path are relatively close.

4. RBUF = 128 KB; path 1: bandwidth = 21 Mbps, and delay is variable; path 2: bandwidth = 21 Mbps, delay = 40 ms.

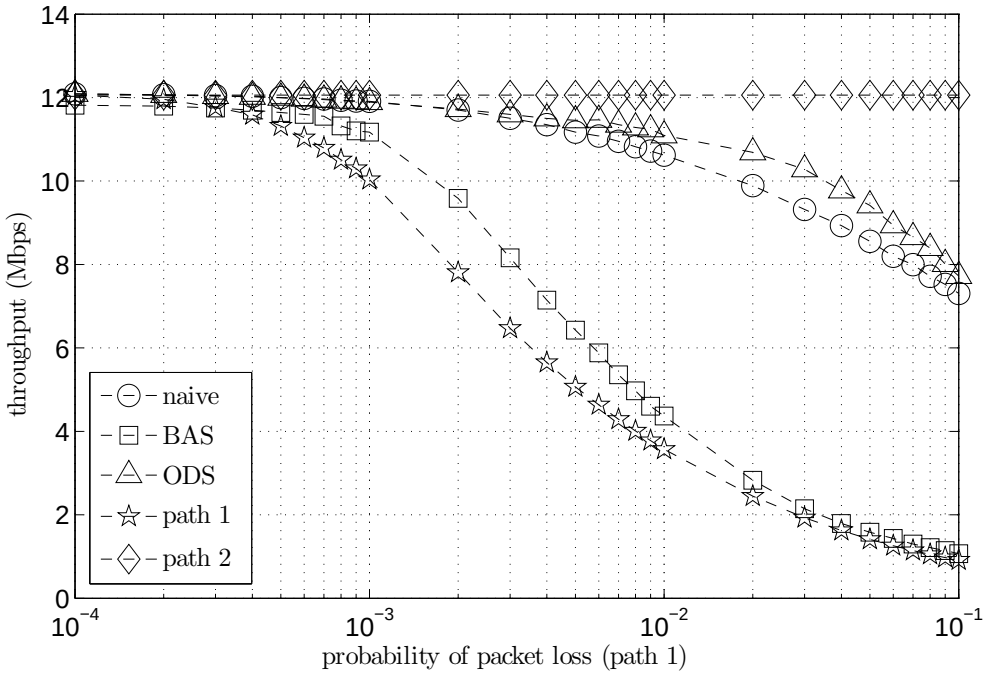


Figure 4.11: Loss-based disparity: throughput results when RBUF is 64 KB.

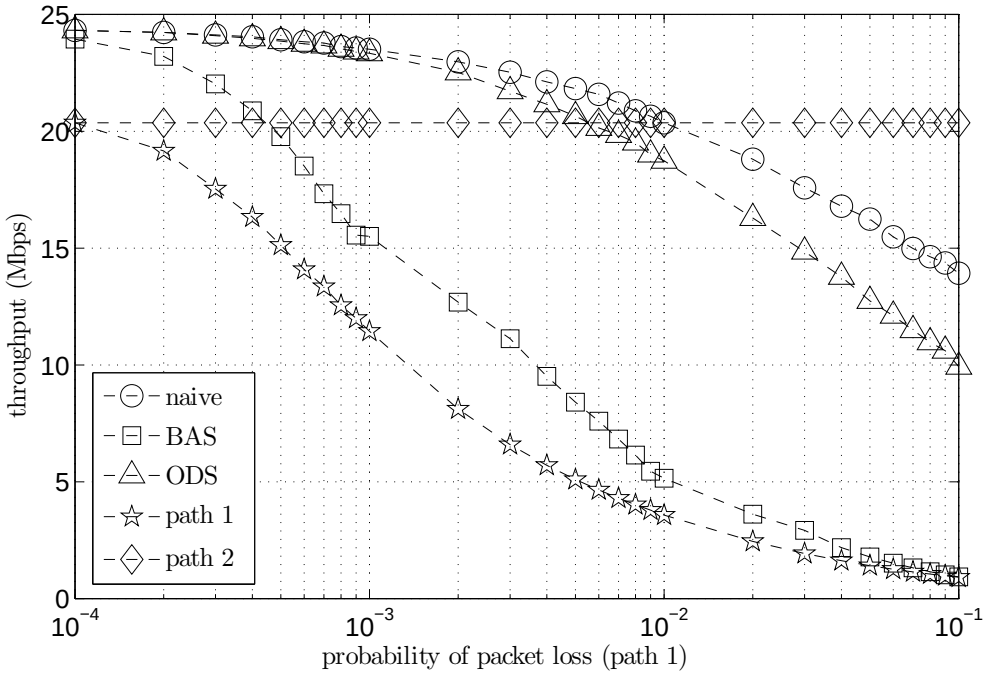


Figure 4.12: Loss-based disparity: throughput results when RBUF is 128 KB.

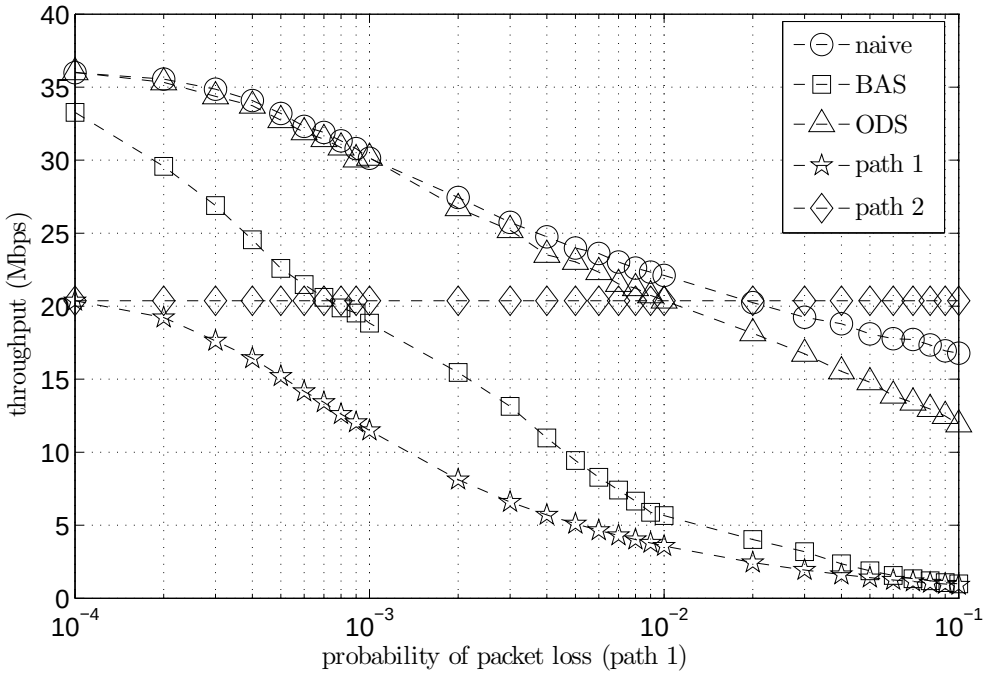


Figure 4.13: Loss-based disparity: throughput results when RBUF is 192 KB.

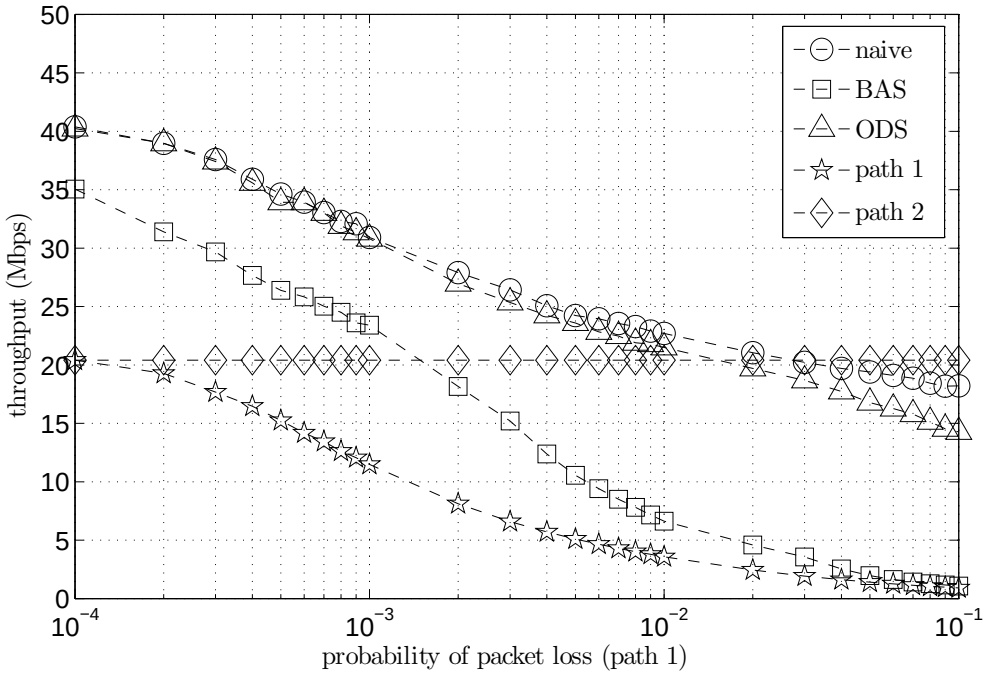


Figure 4.14: Loss-based disparity: throughput results when RBUF is 256 KB.

To get a better understanding of resource utilization, we can approximate how much of the RBUF each path will use (or how many packets need to be in flight) to maximize throughput. Assume the following: $B$ is the available bandwidth of a path (in Bps), $D$ is the propagation delay experienced along a path (in seconds), and $L$ is the size of a packet (in bytes). Given $B$, $D$, and $L$, the number of bytes a path needs to have outstanding (i.e., in flight) to maximize throughput is approximately $B \cdot (L/B + D)$.

With a constant delay of 40 ms, path 2 will always need at least 108 KB in flight to achieve a throughput of 21 Mbps. Alternatively, the delay on path 1 is variable, and when delay is 10 ms, path 1 will only need 27 KB. In turn, this should allow path 2 to have up to 101 KB in flight. Therefore, if 101 KB of data were delivered every 40 ms, then path 2 should achieve a throughput closer to 20 Mbps. But Fig. 4.16 shows throughput at only 9 Mbps. A solution to this problem will be offered in Chapter 6 when we discuss congestion window management.

## 4.4 Summary

To address the issues of receive buffer blocking under delay-based disparity, we proposed the on-demand scheduler. Unlike BAS, our scheduler waits for a transmission opportunity before assigning packets to a destination address. When congestion and flow control allow transmission to a destination, ODS searches the SBUF in a recursive manner, looking for a packet that cannot be delivered to another destination sooner. We tested ODS under three network scenarios: (1) delay-based disparity, (2) bandwidth-based disparity, and (3) loss-based disparity.

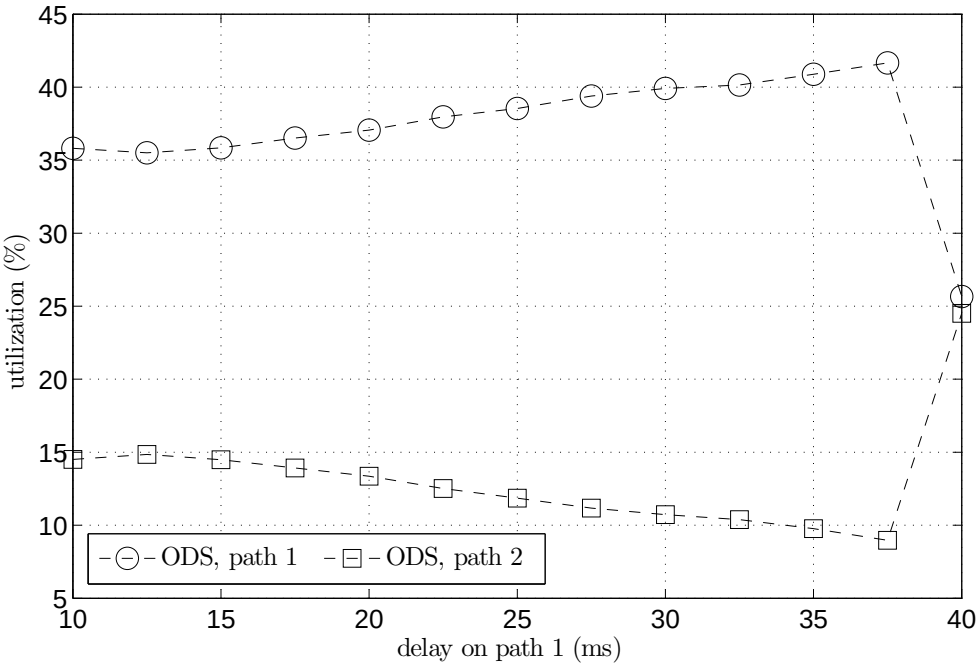


Figure 4.15: Delay-based disparity: destination utilization.

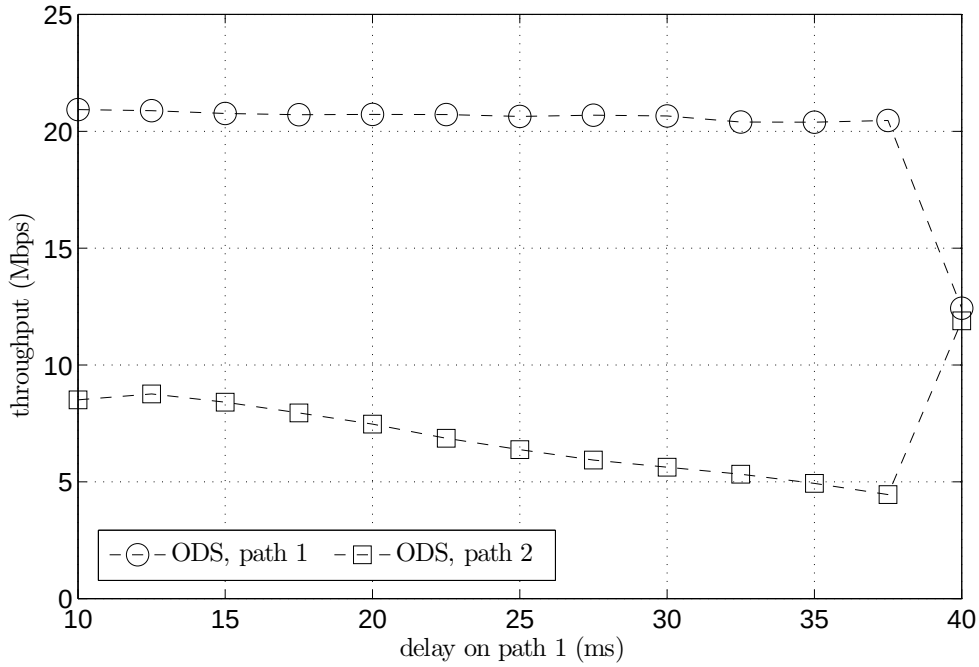


Figure 4.16: Delay-based disparity: destination throughput.

# Chapter 5

# CMT: SCTP Modelling

As an alternative to simulation, mathematical models are often used as a time saving device to study networking protocols. Even though a model is only an approximation, performance results are typically generated in just a fraction of the time simulation can take. In this chapter, we will use two different modelling techniques to develop a method for approximating the throughput of CMT; while one is based on renewal theory the other uses a Markov chain. As we will see, the Markov model is more accurate, but is also more computationally expensive. Alternatively, the renewal model uses a closed-form expression, allowing real-time results albeit accuracy is less consistent.

The rest of the chapter is organized as follows: Section 5.1 provides a brief literature review on transport layer modelling and concurrent multipath transfer; Section 5.2 proposes the respective analytic models; Section 5.3 presents numerical results, comparing the models; finally, Section 5.4 summarizes the work.

## 5.1 Review of Transport Layer Modelling

Even though our work is geared toward SCTP, the models in this chapter are closely related to TCP; so much so, a comprehensive review of TCP literature is warranted[1].

1. Only renewal theory and Markovian models will be evaluated. The interested reader, looking for additional transport layer models, should consult a more thorough treatise on

Arguably, renewal theory is the most popular technique for modelling TCP throughput. The seminal ideas for renewal theory as a TCP throughput model were first discussed by Mathis et al. in [62]. The authors developed a simple model for long-lived TCP connections with periodic independent packet losses. In this work, only linear CWND growth and fast recovery was considered. Moreover, it was assumed timeouts were avoided and the CWND was only ever halved after a loss event. Although loss events were assumed independent, a train of packets following the first loss were also assumed to be lost. Thus, packet losses were considered to be correlated using the implicit assumption of drop-tail queues along the path of a TCP connection. The work in [62] was later cultivated by Padhye et al. [39] to fit a more realistic implementation of TCP Reno. This model assumed both triple duplicate losses as well as timeout events, with exponential back-off periods following a timeout. Other authors again revisited this model to address some inaccuracies and incorporate the slow-start process after timeout events [63,64]. This work is now notoriously known as the PFTK-model[2]. More recently, the PFTK model was transformed to use available bandwidth instead of loss rate as a modelling parameter [65]. The authors argue that bandwidth is a more efficient way to characterize the network in order to approximate system throughput quickly. Other research, based on renewal theory, can be found in [66–69].

Next, using Markov chains and fixed-point methods, a relatively newer source of TCP modelling has evolved [70–72]. Starting from an arbitrary arrival rate, the solution to an M/M/1/K queue provides a Markov chain with average delay and probability of packet loss. Using these parameters in a feedback loop, the Markov

---

the matter, such as [61]. Other modelling approaches include: fluid, processor sharing, and control theoretic models.

2. PFTK is an acronym for the last name initials of all four authors in [39].

chain creates a new arrival rate for the M/M/1/K queue. The model converges by iterating through arrival rates until some minimum error, between input and output, is satisfied. Using this same methodology, Fu et al. [73] developed a model to capture some of the multihoming capabilities featured in SCTP. Later, the same Markov chain from [73] was updated to include the number of losses from a previous round [74]. Although tracking the number of losses adds more states to the Markov chain, the model can now account for the number of transmissions in the interim round between congestion avoidance and fast recovery; results showed improved accuracy for single source transfers. Other TCP models based on Markov chains can be found in [75,76].

## 5.2 Modelling CMT

The objective of the model is to approximate the throughput of a reliable transport layer protocol employing CMT. Strictly speaking, we aim to model the performance of an SCTP session, transferring a large file from a singlehomed sender to a multihomed receiver. Our work, however, should be viewed as supplemental work to traditional transport layer modelling (e.g., a TCP connection with two singlehomed end-points), since great effort has been taken to exploit the well-known techniques of past research ventures.

In order to use a technique, we first need to understand the differences between our system and that to which the technique was previously applied. Fortunately, CMT is mostly an extension to the traditional transport layer connection; instead of one network path separating the sender from receiver, there are several. In addition, each network path has its own set of characteristics (e.g., bandwidth, delay, loss rate), so CMT handles congestion control on a per destination basis. For example, an independent CWND is allocated to each destination address. Sadly, the same type of

demarcation is not so easily applied to flow control as there is only one RBUF. Even if the RBUF were partitioned for each destination, a reliable transport layer protocol must maintain packet order before passing data to the application layer. Therefore, when CMT is employed, no major changes to conventional congestion control are needed, but flow control requires modification. In Chapter 4, however, we used a packet scheduler to address flow control issues and quell receive buffer blocking due to delay-based disparity. Although our packet scheduler could not avoid receive buffer blocking under loss conditions, we hypothesized that an ideal scheduler could predict packet loss; receive buffer blocking would then disappear if a redundant packet was sent simultaneously along an alternative path. For modelling purposes, we can assume such an ideal scheduler so that packets are always received in-order[3].

Finally, unless we specify CWND limits, modelling a shared RBUF is likely to be unmanageable. For example, a Markov chain would need the size of every destination's CWND for each individual state; the state space would then grow exponentially with every additional destination added to the problem domain. In this work we will therefore assume each CWND is limited to some maximum value (preferably its corresponding BDP), where the sum of CWND limits is always less than or equal to the RBUF. Restricting CMT to the aforementioned framework will now allow us to model each network path individually; a remarkably tractable method for modelling CMT.

3. Assuming end-to-end delay never changes, a packet's arrival time would be deterministic. Moreover, predictable packet loss should eliminate receive buffer blocking through redundant transmission.

### 5.2.1 Model Discrepancies

Before presenting our models, we need to highlight the differences between our implementations and those of past research efforts.

#### 5.2.1.1 Markov Model

First, our model is for a single source only, therefore we do not use any queueing theory (e.g., M/M/1/k) nor implement a fixed-point method. The reason behind a single source model lies in our requirements for CMT. Since CMT specifies a shared RBUF, we assume at any given time the sum of CWNDs must be less than or equal to the size of the RBUF. Later in Chapter 6, we will show how to optimize performance by carefully managing the size of each CWND. Unfortunately, the solution to a queue model with multiple sources cannot offer loss rate or delay information unless every source (sharing the network path) is identical. Moreover, if we want CMT to manage CWNDs on a destination basis, we cannot expect every source to have the same statistical properties (e.g., average CWND); therefore, we must use a single source model in this endeavour.

Another difference is an increase in state-space. In previous models, the Markov chain predicts fast recovery (FR) and timeout (TO) events during congestion avoidance rounds that have yet to experience packet loss. In this way, the model will predict a FR event immediately following a round with losses. While this is not impossible, it is unlikely, and if a correlated loss model is used (e.g., one that assumes if a packet is lost all remaining packets transmitted in the same round are also lost), this is most certainly untrue. Furthermore, if the model jumps from a round without losses to a round that has already invoked FR, and thus the CWND is reduced by

a factor of 2, the accuracy of the model suffers because the number of packets that would have been transmitted in an interim round will go unnoticed.

#### 5.2.1.2 Renewal Model

For the most part, our renewal model is derived using a method similar to past TCP implementations, with the exception of TO probability and the exponential back-off period. Since SCTP requires 4 SACKs to trigger a TO, compared to TCP's 3 duplicate acknowledgements, the formula for the probability of a TO event needs to be modified. Previous TCP models, moreover, limit the doubling of RTO to 7 iterations, without specifying a maximum. We, on the other hand, assume $\text{RTO}_{\text{max}}$ is given; so RTO can be doubled any number of times.

### 5.2.2 Basic Modelling Assumptions

Given a multihomed network topology (see Chapter 3), we assume a set of network paths connect an SCTP sender/receiver pair, so that the sender may transmit over any path and all successful packets will arrive at the receiver. Each network path, moreover, is defined by its bandwidth (i.e., maximum transmission rate), probability of packet loss, and additional packet delay (made up from other delay sources like propagation, processing, and queueing effects). What is more, we assume a path to be independent of all other paths as well as its characteristics to remain constant.

Similar to past research, we also model the behaviour of our system in terms of discrete rounds. A round starts with the transmission of packets, and ends with the reception of a SACK. Depending on the state of the system, the length of time between rounds may take as little as one RTT, one RTO, or up to the maximum allowable retransmission timeout (i.e., $\text{RTO}_{\text{max}}$). During a round, moreover, packets can be

lost, where packet losses are characterized by assuming one of two loss models: (1) independent, or (2) correlated [62]. An independent loss model, for example, assumes each packet transmitted in a round will have the same probability of being lost, regardless of when it is transmitted during the round. An independent loss model can be used when links are lossy (e.g., over a wireless channel). The probability of losing $\gamma$ packets when $\xi$ are transmitted under an independent loss model is given by

$$P(\gamma, \xi) = \binom{\xi}{\gamma} p^{\gamma} (1-p)^{\xi-\gamma}, \tag{5.1}$$

where $p$ is the probability of packet loss on the network path.

Alternatively, a correlated loss model assumes a packet is lost with probability $p$ as long as no other packet has been lost in the same round. Assuming a packet has already been lost, however, all remaining packet transmissions are lost with probability 1. This loss model represents a drop-tail queue where packets are lost when a bottleneck queue reaches capacity, thereby dropping any packets arriving after capacity has been reached. Under the correlated loss model, the probability of losing $\gamma$ packets when $\xi$ are transmitted is calculated by

$$P(\gamma, \xi) = \begin{cases} (1-p)^{\xi}, & \gamma = 0 \\ p(1-p)^{\xi-\gamma}, & 0 < \gamma \leq \xi \\ 0, & \text{otherwise.} \end{cases} \tag{5.2}$$

Packet RTTs are calculated using the following equation

$$\text{RTT} = \begin{cases} d + \frac{1}{b}, & \text{if } b \cdot d \geq \text{CWND} \\ \frac{\text{CWND}}{b}, & \text{otherwise,} \end{cases} \tag{5.3}$$

where $b$ is the bandwidth of a bottleneck link (in packets/second), and $d$ is a constant delay experienced along the path. By assuming delay to be constant, including queueing delay from background traffic sources, at least the first case of Eq. (5.3) should be considered adequate. With that said, queueing delay from the same source can vary; for example, when the number of transmissions exceeds the BDP, packets will be backlogged as the path becomes congested. This creates a queueing effect and increases RTT. If the number of packets in the queue, $q$, were known ahead of time, we would have

$$\text{RTT} = d + \frac{q}{b}. \tag{5.4}$$

Even if $q$ is unavailable, we know that acknowledgements should start arriving at regular intervals when the CWND is greater than the BDP. When this happens, a packet leaves the path every $1/b$ seconds. Assuming a packet leaves the moment a new round begins, another CWND$/b$ iterations will occur before the packet that ends the round leaves the path.

Finally, we assume the variation in RTT to be RTT/2. Therefore the initial RTO will always be $3 \cdot \text{RTT}$ since

$$\text{RTO} = \text{RTT} + 4 \cdot \text{Var}[\text{RTT}]. \tag{5.5}$$

### 5.2.3 Markov Model

Our first model uses a discrete-time Markov chain (DTMC) to model the behaviour of a network path; the goal of which is to generate a state-transition probability matrix $\mathbf{Q}$, and solve the steady-state probability distribution $\boldsymbol{\pi}$ for the DTMC.

We first define a state to be a round where the CWND is $\omega$ packets, $\xi$ packets will be transmitted, and $\tau$ is the current SSTHRSH. At the end of a round, the

system transitions from state $i$ to $i'$, or $(\xi, \gamma, \tau)$ to $(\xi', \gamma', \tau')$, where the transition probability is defined by element $Q(i, i')$ and the steady-state probability distribution is computed as

$$\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{Q}. \tag{5.6}$$

Next, and for the sake of convenience, we will classify and group each state into one of the following descriptive subsets: Congestion Avoidance (CA), Slow-start (SS), and Exponential Back-off (EB). Finally, we formulate a throughput expression based on the steady-state probability.

#### 5.2.3.1 Congestion Avoidance

We begin our rendering of CA mode by defining the set of CA states as

$$\mathcal{C} = \{(\omega, \xi, 1) : 2 \leq \omega \leq \omega_{\max}, 1 \leq \xi \leq \omega, \omega \in \mathbb{Z}, \xi \in \mathbb{Z}\}, \tag{5.7}$$

where $\xi_{\max}$ is the size of the RBUF in packets[4].

During CA mode, transition probabilities are dependent on $\omega$, $\xi$, and $\gamma$. If $\omega = \xi$, however, we need only to concern ourselves with $\omega$ and $\gamma$ since it means that no packets were lost in the previous round; allowing us to ignore FR events. Assuming that $\omega = \xi$ and $\gamma < \omega$, the system will stay in CA, and after one RTT will transition from $(\omega, \xi, 1)$ to $(\omega', \xi', 1)$ with probability $P(\gamma, \omega)$, where $\omega' = \omega$ and

$$\xi' = \begin{cases} \omega - \gamma, & \text{if } \gamma \geq 1 \\ \omega + 1, & \text{if } \gamma = 0, \omega < \omega_{\max} \\ \omega, & \text{otherwise.} \end{cases} \tag{5.8}$$

4. State (1,1,1) is reserved for EB.

Alternatively, if $\gamma = \omega$, no packets will be acknowledged and a TO event will send the system into EB mode. In this case, the system will transition to $(1, 1, 2^{\lfloor \log_2 \omega/2 \rfloor})$ with probability $P(\omega, \omega)$ in one RTO.

If $\omega > \xi$, then packets must have been lost in the previous round. Whether or not the system stays in CA is now conditional on $\omega$ and $\gamma$ as well as $\xi$. Given that $\omega > \xi$, the system will stay in CA mode only if a FR event is triggered. To invoke a FR event, the sender must receive four out-of-order SACKs in the next round. Regardless of $\gamma$ and $\omega$, as long as $\xi < 4$, a TO is inevitable and the system will transition into state $(1, 1, 2^{\lfloor \log_2 \omega/2 \rfloor})$ with a probability of 1 after one RTO.

Even if $\xi \geq 4$, however, there remains two conditions for which a TO event can still occur. The first is intuitive, if less than 4 packets are acknowledged, so that $\xi - \gamma < 4$, a TO event occurs simply because a FR did not. The second condition, on the other hand, happens when too many packets are lost in the current round, so that the number of outstanding packets is greater or equal to the size of the CWND after FR. Following a FR event, the CWND is divided by 2, so the number of packets transmitted in the next round will be $\omega/2 - \gamma$. Clearly, if $\gamma \geq \omega/2$, then no packets will be transmitted in the next round and a resulting TO event will occur. Assuming that $\omega > \xi$ and $\xi \geq 4$, after the duration of one RTT the system will transition to $(\omega/2, \omega/2 - \gamma, 1)$ with probability $P(\xi, \gamma)$ as long as $\xi - \gamma > 3$ and $\omega/2 - \gamma > 0$; otherwise, after one RTO it will transition to $(1, 1, 2^{\lfloor \log_2 \omega/2 \rfloor})$ with probability

$$\sum_{i=1}^{\xi} P(\xi, i), \quad \text{if } i \geq \xi - 3 \text{ or } i \geq \omega/2. \tag{5.9}$$

#### 5.2.3.2 Slow-start (SS)

A SS state exists when the CWND is less than its SSTHRSH (i.e., $\omega < \tau$). During a SS round, moreover, every CUMACK increases the CWND by one MTU; so if no packets are lost, the CWND is doubled between SS rounds. The set of SS states for our Markov model is defined as

$$\mathcal{S} = \{(\omega, \xi, \tau) : \omega = \xi = 2^i, 1 \leq i \leq \log_2(\tau), \tau \in \mathcal{T}, \tau > 1, i \in \mathbb{Z}\}, \quad (5.10)$$

where $\mathcal{T}$ is the set of slow-start thresholds, given by

$$\mathcal{T} = \{2^i : 0 \leq i \leq \lfloor \log_2(\xi_{\max}/2) \rfloor, i \in \mathbb{Z}\}. \quad (5.11)$$

Assuming that $\omega < \tau$ and $\gamma = 0$, the system will either remain in SS or move into CA mode, depending on the values of $\omega$ and $\tau$. If $\tau > 2\omega$, the system will stay in SS and transition to $(2\omega, 2\xi, \tau)$; otherwise, the system will start CA mode and transition into $(2\omega, 2\xi, 1)$. In either case, the transition probability will always be $P(0, \omega)$, and the duration of time between transitions will be one RTT.

On the other hand, if $\gamma > 0$, the system can either remain in SS until a FR, or TO and move into EB. In an attempt to reduce state space, when $1 \leq \gamma < \omega$ we move the system into CA by letting $\tau' = 1$. Although this simplifies the model, we believe this should have little if no impact on throughput since a pending FR (in the following rounds) would move the system into CA anyways. Therefore, assuming $1 \leq \gamma < \omega$, the system will transition to $(2\omega - \gamma, 2(\omega - \gamma), 1)$ with probability $P(\gamma, \omega)$ after one RTT.

Finally, if $\gamma = \omega$, the system will TO and move into EB by transitioning to $(1, 1, 2^{\lfloor \log_2 \omega/2 \rfloor})$ with probability $P(\omega, \omega)$ after one RTO.

#### 5.2.3.3 Exponential back-off

EB mode starts immediately following a TO event and continues until a packet is successfully delivered to the receiver. We define the set of EB states by

$$\mathcal{E} = \{(0, \xi, \tau) : (2 \leq \xi \leq M, \tau = 1) \cup (\xi = 1, \tau \in \mathcal{T})\}, \tag{5.12}$$

where $M$ is the maximum number of consecutive TO events, calculated as

$$M = \max\left(1, \left\lfloor \log_2 \frac{\text{RTO}_{\max}}{\text{RTO}} \right\rfloor\right). \tag{5.13}$$

To understand our definition of $\mathcal{E}$, we actually redefine $\xi$ to mean the number of consecutive TO events instead of the number of packets that are sent in a round. Although this is unorthodox, we should mention that the number of packets transmitted during an EB round will always equal the size of the CWND (i.e., $\omega = 1$). What changes, rather, is the time between consecutive EB rounds. Following every TO event, the RTO is doubled until RTO reaches some maximum value (i.e., $\text{RTO}_{\max}$). Later on, it will become necessary for us to generate an approximation for the duration of time between rounds; and since the time between EB rounds doubles after every consecutive TO event, we will need to know the probability of being in any particular EB round.

Since we only need to consider the transmission of one packet, the transition probability out of an EB state will always be $P(0, 1)$, with a round time of one RTT. In fact, the system will always transition to $(2, 2, \tau)$ after a successful packet delivery[5]. Furthermore, as long as $M > \xi + 1$, a failed transmission attempt will always transition the system into $(1, \xi + 1, 1)$ with probability $P(1, 1)$ after $2^{\xi}\text{RTO}$.

5. In this case we are assuming $\xi_{\max} > 1$.

But if $M = \xi + 1$ or $M = \xi$, another TO event will keep the system in the same EB state, $(1, M, 1)$, every $\text{RTO}_{\max}$.

#### 5.2.3.4 Throughput

In this model we express throughput as the number of packets received per unit time, denoted by $\eta$. The average throughput of a source will then be

$$\eta = \frac{E[\xi] - E[\gamma]}{E[\delta]}, \tag{5.14}$$

where $E[\xi]$ is the expected number of packets transmitted per round, $E[\gamma]$ is the expected number of packets lost per round, and $E[\delta]$ is the expected duration of time between rounds.

The expected number of packets transmitted per round is given by

$$E[\xi] = \sum_{i \in \{\mathcal{C}, \mathcal{S}\}} \pi(i)\xi(i) + \sum_{i \in \mathcal{E}} \pi(i), \tag{5.15}$$

where $\pi(i)$ returns the probability of being in state $i$ and $\xi(i)$ is the number of packets transmitted in state $i$.

Using either Eq. (5.1) or (5.2), the expected number of packets lost per round can be expressed by

$$E[\gamma] = \sum_{i \in \{\mathcal{C}, \mathcal{S}\}} \pi(i) \sum_{j=1}^{\xi(i)} jP(j, \xi(j)) + \sum_{i \in \mathcal{E}} \pi(i)P(1, 1). \tag{5.16}$$

Before we provide our expression for $E[\delta]$, we first define $\mathcal{D}$ to be a matrix of round times (i.e., the time between state transitions), and $\mathcal{A}$ to be the set of all states (i.e., $\mathcal{A} = \{\mathcal{C}, \mathcal{S}, \mathcal{E}\}$). Finally, the expected duration of time between rounds is

calculated by

$$E[\delta] = \sum_{i \in \mathcal{A}} \sum_{i' \in \mathcal{A}} \pi(i) D(i, i'), \tag{5.17}$$

where $D(i, i')$ is an element of $\mathcal{D}$ representing the time it takes to transition from state $i$ to $i'$.

### 5.2.4 Renewal Model

Our second model uses renewal theory: an assumption that a stochastic process continually restarts at regular intervals. When we consider the additive increase/multiplicative decrease (AIMD) algorithm used for congestion control, a continuous sawtooth pattern begins to form. Our goal, therefore, is to represent this pattern as an average interval in order to formulate a closed-form expression for the throughput of a single network path.

By letting $S_t$ and $L_t$ be the number of packets transmitted and lost in the time interval $[0, t] : t > 0$, we can express the average throughput of an SCTP session by

$$\eta = \lim_{t \to \infty} \frac{S_t - L_t}{t}, \tag{5.18}$$

or

$$\eta = \frac{E[S] - E[L]}{E[T]}, \tag{5.19}$$

where $E[S]$, $E[L]$, and $E[T]$ are the expected packets sent, the expected packets lost, and the expected time of an interval, respectively.

Similar to the Markov model, the renewal model is also broken into three descriptive modes of transmission: congestion avoidance (CA), exponential back-off (EB), and slow-start (SS).

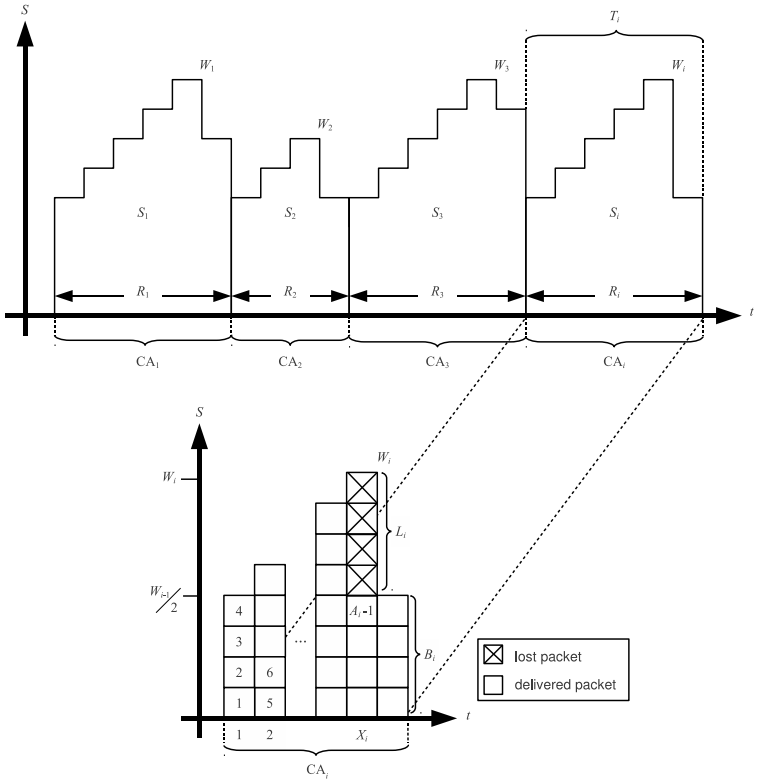


Figure 5.1: A continuous series of congestion avoidance periods.

#### 5.2.4.1 Congestion Avoidance Mode

We begin our discussion of CA mode by focusing our attention on Fig. 5.1; a depiction of a continuous series of CA periods. Each CA period consists of $R$ transmission rounds where $S$ packets are sent and $L$ packets lost. Using renewal theory, our goal is to approximate the throughput of the $i^{\text{th}}$ CA period. During $\text{CA}_i$, new packets are sent every RTT, where the number of packets transmitted equals the current CWND, $W$. If no packets are lost, $W$ increases until some maximum window is reached. Following a round of losses, however, $W$ remains unchanged but in the next round the number of packets transmitted equals $W - L$. Typically, a CA period ends with FR and the process begins all over again with a CWND of $W/2$ packets.

We observe that $\text{CA}_i$ will always begin with $W_i = W_{i-1}/2$, increasing by one

packet every round thereafter. The number of packets transmitted during $\mathrm{CA}_i$ can now be expressed by

$$\begin{aligned} S_i^{\mathrm{CA}} &= \sum_{k=0}^{X_i-1} \left( \frac{W_{i-1}}{2} + k \right) + B_i \\ &= \frac{X_i}{2}(W_{i-1} + X_i - 1) + B_i, \end{aligned} \tag{5.20}$$

where $B_i$ is the number of packets sent in a round following a loss.

Now if we let $A_i$ be the send count when the first loss occurs and assume a correlated loss model, we can also write

$$S_i^{\mathrm{CA}} = A_i + W_i - 1. \tag{5.21}$$

By assuming a correlated loss model, packets are (successfully) received according to a geometric distribution. Therefore, the probability that $A_i = k$ packet transmissions can easily be calculated by $p(1-p)^{k-1}$. So the expected send count after the first loss will simply be

$$E[A] = \sum_{k=1}^{\infty} pk\,(1-p)^{k-1} = \frac{1}{p}. \tag{5.22}$$

From (5.21) and (5.22) it follows that

$$E[S^{\mathrm{CA}}] = \frac{1-p}{p} + E[W]. \tag{5.23}$$

Since the CWND at the end of $\mathrm{CA}_i$ can be written as

$$W_i = \frac{W_{i-1}}{2} + X_i - 1, \tag{5.24}$$

when using (5.24) in (5.20), we get

$$S_i^{\mathrm{CA}} = \frac{X_i}{2}\left(\frac{W_{i-1}}{2} + W_i\right) + B_i. \tag{5.25}$$

By assuming $\{X_i\}$ and $\{W_i\}$ to be mutually independent sequences with i.i.d. random variables, equating (5.23) and (5.25) yields

$$\frac{1-p}{p} + E[W] = \frac{E[X]}{2}\left(\frac{E[W]}{2} + E[W] - 1\right) + B_i. \tag{5.26}$$

Applying this assumption to (5.24) also gives

$$\begin{aligned} E[W] &= \frac{E[W]}{2} + E[X] - 1 \\ &= 2E[X] + 1. \end{aligned} \tag{5.27}$$

Furthermore, if we assume $B_i$ to be uniformly distributed between 1 and $W_{i-1}$, $E[B]$ simplifies to $E[W]/2$. Likewise, since $L_i = W_i - B_i$, then $E[L]$ will also be $E[W]/2$. So solving (5.26) for E[X], then substituting into (5.27) yields

$$E[W] = \sqrt{\frac{8(1-p)}{3p} + \frac{1}{9}} - \frac{1}{3}. \tag{5.28}$$

Similarly, we will also have

$$E[X] = \sqrt{\frac{2(1-p)}{3p} + \frac{1}{36}} + \frac{5}{6}. \tag{5.29}$$

Finally, since the number of transmission rounds in $\mathrm{CA}_i$ will always be

$$E[R] = E[X] + 1, \tag{5.30}$$

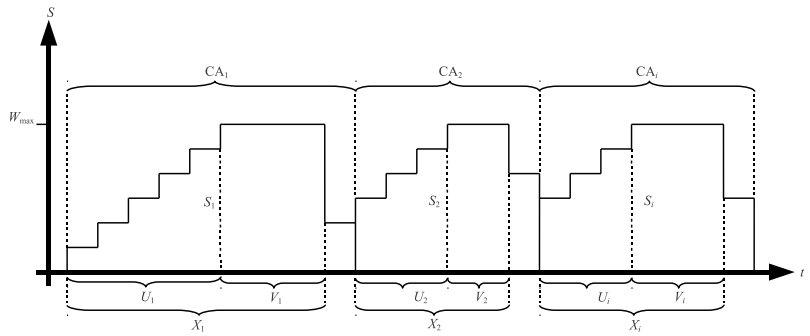


Figure 5.2: Evolution of the CWND constrained by $W_{\max}$.

the expected duration of $\mathrm{CA}_i$ is just

$$E[T] = E[R] \cdot \mathrm{RTT}. \tag{5.31}$$

Before moving on, however, we need to address the possibility of a constrained RBUF. During connection setup, the receiver advertises a maximum buffer size, which ultimately determines the sender's maximum CWND, $W_{\max}$; so when $W = W_{\max}$, the send count (per round) stays the same. With respect to Fig. 5.2, we will now describe how to model this effect. During the first CA period, the CWND grows linearly from 1 to $W_{\max}$ for $U_1$ rounds, then remains constant for $V_1$ rounds until a FR event. Finally, the CWND drops to $\frac{W_{\max}}{2}$, the sender recovers, and the process repeats, only this time starting from $\frac{W_{\max}}{2}$, so

$$W_{\max} = \frac{W_{\max}}{2} + U_i - 1. \tag{5.32}$$

The calculation for the number of packets sent during $\mathrm{CA}_i$ will now have the form

$$S_i^{\mathrm{CA}} = \frac{U_i}{2}\left(\frac{W_{\max}}{2} + W_{\max}\right) + V_i W_{\max} + B_i. \tag{5.33}$$

Substituting (5.32) into (5.33), moreover, gives

$$E[S^{\mathrm{CA}}] = \frac{3}{8}W_{\max}^2 + W_{\max}E[V] + \frac{1}{4}W_{\max}. \tag{5.34}$$

Then using (5.23) and solving for $E[V]$ yields

$$E[V] = \frac{1-p}{pW_{\max}} - \frac{3}{8}W_{\max} + \frac{3}{4}. \tag{5.35}$$

From Fig. 5.2, we see that $X_i = U_i + V_i$, so

$$E[X] = \frac{1-p}{pW_{\max}} + \frac{W_{\max}}{8} + \frac{3}{4}. \tag{5.36}$$

We now have two throughput expressions for CA mode: (1) when $E[W] < W_{\max}$, and (2) when $E[W] \geq W_{\max}$[6].

#### 5.2.4.2 Exponential Back-off

Every time a new round begins, the sender starts a timer and waits up to one RTO to receive an acknowledgement. If this timer expires before receiving an acknowledgement, the sender triggers a TO event. Following a TO, the sender enters exponential back-off (EB) mode, dropping the CWND to one and doubling RTO. This process is continued for every subsequent TO until reaching $\mathrm{RTO}_{\max}$, where it remains constant until a packet is finally acknowledged. Fig. 5.3 illustrates a cycle of consecutive CA periods followed by one EB mode.

6. The informed reader will notice this is the same result as the PFTK model [39].

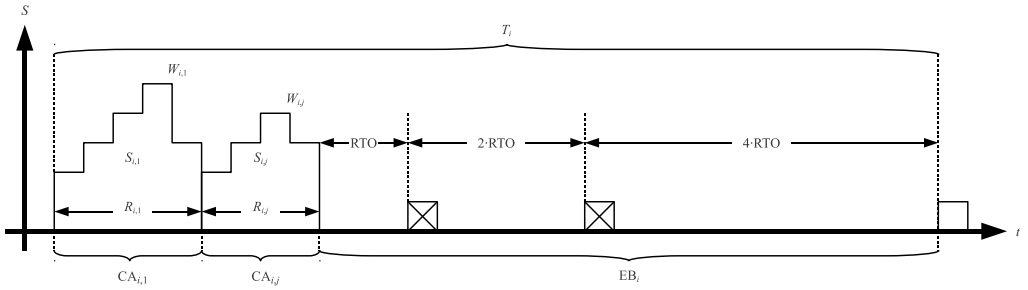


Figure 5.3: Evolution of the transport layer during CA and EB periods.

Translating Fig. 5.3 into a mathematical representation of throughput will yield

$$\eta = \frac{E[S^{\mathrm{CA}}] - E[L^{\mathrm{CA}}]}{E[T^{\mathrm{CA}}] + P^{\mathrm{TO}} E[T^{\mathrm{EB}}]}. \tag{5.37}$$

In (5.37), $E[S^{\mathrm{CA}}]$, $E[L^{\mathrm{CA}}]$, and $E[T^{\mathrm{CA}}]$ are the expected number of packets transmitted, the expected number of packets lost and the length of time during CA mode, respectively; $E[T^{\mathrm{EB}}]$ is the duration of an EB period; and $P^{\mathrm{TO}}$ is the probability of a TO event.

There are now two possible outcomes at the end of a CA period: FR or TO. To invoke a FR event, the sender must receive at least four acknowledgements no more than two rounds after the first loss round; otherwise a TO will occur. Given the certainty of at least one loss, we can express the probability that the first $j$ packets are acknowledged after $W$ were sent as

$$P(j, W) = \frac{p(1-p)^j}{1-(1-p)^W}. \tag{5.38}$$

If we now apply (5.38) to all scenarios that would end a CA period, we can calculate

the probability of a TO event by

$$P^{\mathrm{TO}} = \sum_{j=0}^{3} P(j,W) + \sum_{j=4}^{W-1} P(j,W) \sum_{k=0}^{3} p(1-p)^k$$
$$= \frac{1-(1-p)^8-(1-(1-p)^4)(1-p)^W}{1-(1-p)^W}. \tag{5.39}$$

We will now calculate the average duration of an EB period. Recalling that RTO is doubled for each unsuccessful retransmission until some maximum RTO is reached, the duration of $j$ consecutive TO events is expressed by

$$T_j^{\mathrm{TO}} = \begin{cases} (2^{j-1})\mathrm{RTO}, & j \leq M \\ (j-M)\mathrm{RTO}_{\max}, & j > M, \end{cases} \tag{5.40}$$

where $M$ is solved using Eq. (5.13). In this case, however, we are constrained by $\mathrm{RTO}_{\max} \geq 2\mathrm{RTO}$. The expected duration of an EB period would then be

$$E[T^{\mathrm{EB}}] = \sum_{j=1}^{\infty} T_j^{\mathrm{TO}}(1-p)p^{j-1}$$
$$= \frac{(1-p)(1-2^M p^M)}{1-2p}\mathrm{RTO} + \frac{p^M}{1-p}\mathrm{RTO}_{\max}. \tag{5.41}$$

#### 5.2.4.3 Slow-start

Following EB, the sender will increase its CWND according to slow-start (SS) mode, i.e., the number of packets transmitted per round will double every RTT. With reference to Fig. 5.4, the number of packets transmitted in SS period $i$ is calculated by

$$S_i^{\mathrm{SS}} = 1 + 2 + 2^2 + \cdots + 2^{H_i-1} = \sum_{k=1}^{H_i} 2^{k-1} = 2^{H_i} - 1, \tag{5.42}$$

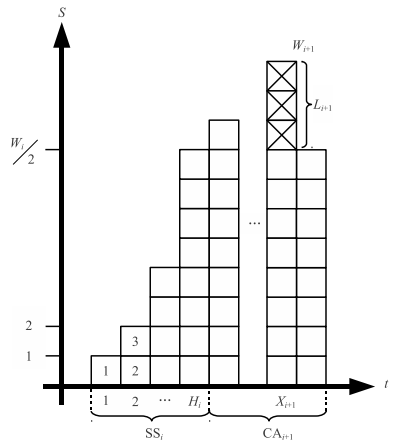


Figure 5.4: Packets sent during slow-start.

where $H_i$ is the number of SS rounds before CA resumes. $H_i$ is easily solved by rearranging (5.42) so that

$$H_i = \log_2 \left( S_i^{\mathrm{SS}} + 1 \right). \tag{5.43}$$

Assuming the last term in (5.42) should be $W_i/2$, we can write

$$W_i = 2^{H_i}. \tag{5.44}$$

Then substituting (5.44) into (5.42) gives

$$S_i^{ss} = W_i - 1, \tag{5.45}$$

and (5.45) into (5.43) yields

$$H_i = \log_2 W_i. \tag{5.46}$$

Since there must be at least one SS round, (5.46) is rewritten as

$$H_i = \max(1, \log_2 W_i). \tag{5.47}$$

| | | |
|---|---|---|
| application layer | continuous file transfer | 60 minutes |
| SCTP end-points | packet size | 1500 bytes |
| | SBUF | $\infty$ |
| | RBUF | 32 - 192 KB |
| | $\text{RTO}_{\max}$ | 60 seconds |
| | RTO parameters | $\alpha = 0.125, \beta = 0.25$ |
| network paths | bandwidth | 10 - 30 Mbps |
| | propagation delay | 10 - 100 ms |
| | prob. packet loss | $10^{-4}$ - $10^{-1}$ |

Table 5.1: Model Comparison: simulation parameters.

From (5.45) we can write the expected number of packets transmitted during slow-start as

$$E[S^{\text{SS}}] = E[W] - 1. \tag{5.48}$$

Finally, the expected duration of SS mode will be

$$E[T^{\text{SS}}] = \max(1, \log_2 E[W]) \cdot \text{RTT}. \tag{5.49}$$

Combining each of SCTP's transmission modes, the final throughput equation using renewal theory will be

$$\eta = \frac{E[S^{\text{CA}}] - E[L^{\text{CA}}] + P^{\text{TO}} E[S^{\text{SS}}]}{E[T^{\text{CA}}] + P^{\text{TO}}(E[T^{\text{EB}}] + E[T^{\text{SS}}])}. \tag{5.50}$$

## 5.3 Model Comparison

We will now compare the accuracy of either model with simulated results[7]. Previously, we established a framework allowing each network path to be modelled independently,

7. Simulated results were generated using *ns*-2.

therefore we only need to use the singlehomed network topology when evaluating accuracy. Later in Chapter 6, we will see how our models can be used for CMT, when we apply them to the problem of congestion window management.

### 5.3.1 Simulation and Model Parameters

We varied the following four parameters in order to generate a range of scenarios: probability of loss event ($p$), bandwidth ($b$), delay ($d$), and receive buffer size ($r$). With that said, simulation parameters were focused around $p = 0.1$ %, $b = 21$ Mbps, $d = 40$ ms, and $r = 128$ KB. We chose these parameters for the following reasons:

**(1)** Measurement studies have produced a wide range of statistics, but little consensus on average loss rates [77–79]. In our work, we assume lower loss rates for ideal conditions.

**(2)** Even though 4G LTE is currently being rolled out, the best available service for most of North America is still only evolved high speed packet access (HSPA+). Under ideal conditions, HSPA+ can offer speeds as high as 21 Mbps.

**(3)** We compared packet RTTs from the University of Western Ontario to a number of other institutions (e.g., the University of Toronto, Princeton University, the University of California, Berkeley). While RTTs varied anywhere between 10 and 100 ms, we decided on a delay of 40 ms because this was the average among institutions within our timezone.

**(4)** A minimum RBUF of 128 KB for an SCTP session employing CMT was recommended in [34].

Unless otherwise specified, Table 5.1 lists all other simulation parameters.

| Plot | Model | Accuracy (%) | | | Accuracy (Mbps) | | |
|---|---|---|---|---|---|---|---|
| | | max | min | mean | max | min | mean |
| Fig. 5.5 | renewal | 12.7 | 0.063 | 2.35 | 2.30 | ∼0 | 0.32 |
| | Markov | 1.24 | 0.098 | 0.69 | 0.21 | 0.003 | 0.05 |
| Fig. 5.6 | renewal | 12.0 | 0.16 | 3.9 | 1.08 | 0.009 | 0.37 |
| | Markov | 1.23 | 0.69 | 0.89 | 0.094 | 0.054 | 0.083 |
| Fig. 5.7 | renewal | 14.7 | 0.49 | 3.37 | 2.56 | 0.067 | 0.37 |
| | Markov | 1.11 | 0.01 | 0.45 | 0.23 | 0.008 | 0.047 |
| Fig. 5.8 | renewal | 6.78 | 0.73 | 3.67 | 0.64 | 0.075 | 0.35 |
| | Markov | 4.05 | 0.26 | 1.09 | 0.22 | 0.027 | 0.092 |

Table 5.2: Model Comparison: statistical accuracy with simulated results.

### 5.3.2 Results and Discussion

The results of our comparison are shown in Figs. 5.5 - 5.8. While in all scenarios the Markov model proved most accurate, renewal theory did a reasonable job despite some areas of deviation. For example, in Fig. 5.7 the renewal model overestimated throughput by a margin of at most 14.7% (or 2.6 Mbps), but percent difference never exceeded 1.11% (or 0.23 Mbps) when using the Markov model. See Table 5.2 for a complete list of statistical results for each test.

Since the Markov model takes all considerations into account, its pinpoint accuracy should come as no surprise. Therefore, if model parameters were to be sampled in real-time, the Markov model should guarantee a good estimate on throughput potential. Unfortunately, since the Markov model needs to solve a system of linear equations, it suffers from scalability issues. For instance, Fig. 5.8 shows us the limitations of the Markov model; when the RBUF is 176 KB, the state-space becomes so large that we can no longer solve for the steady-state probability distribution[8].

Even with more memory, however, the Markov model will still face longer pro-

8. The Markov model was implemented using MATLAB R2009b 32-bit on a Linux desktop computer with 3 GB of RAM.

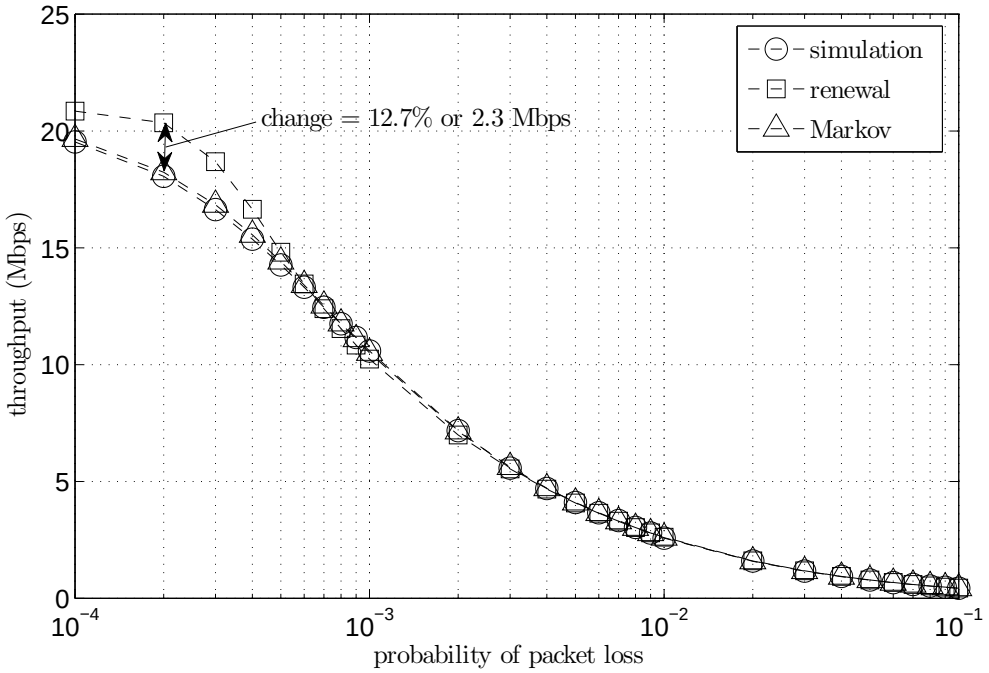


Figure 5.5: Model comparison: $b = 21$ Mbps, $d = 40$ ms, RBUF = 128 KB.

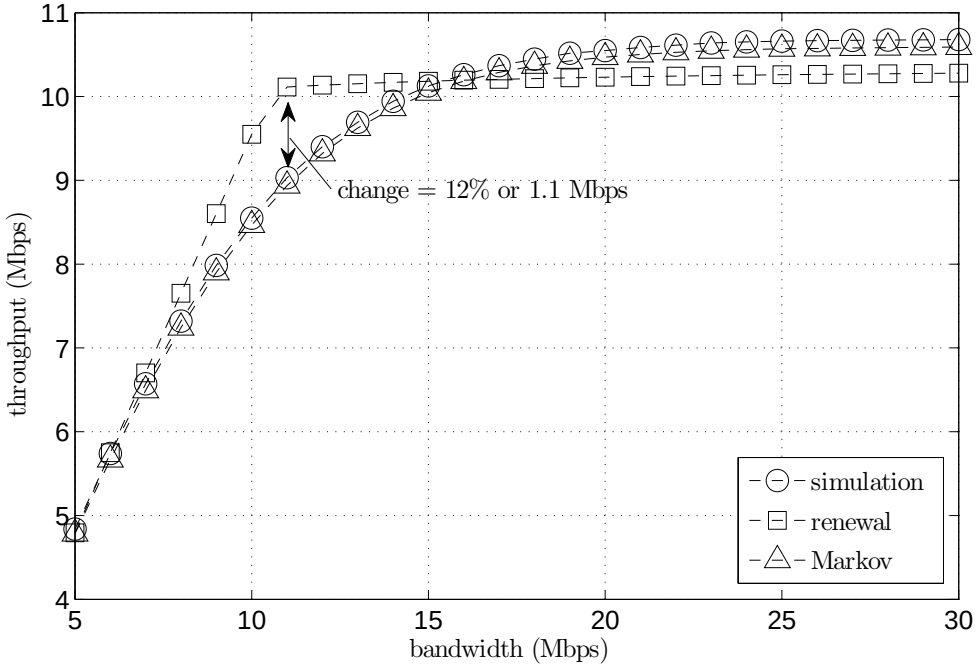


Figure 5.6: Model comparison: $p = 10^{-3}$, $d = 40$ ms, RBUF = 128 KB.

cessing delays as the number of states increase. To put this into context, $\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{Q}$ is typically solved using Gaussian Elimination (GE), where multiplication and subtraction operations are considered units of computation. Assuming there are $n$ equations, the total number of operations on the left side of the system of equations require $\frac{1}{3}(n^3 - n)$, while the right side needs $n^2$ [80]; giving GE a computational complexity of $\mathcal{O}(n^3)$. In the Markov model, $n$ represents the total number of valid states, i.e., all CA, SS, and EB states. Since the number of CA and SS states are dependant on $\omega_{\max}$, $n$ increases with $\omega_{\max}$[9]. In addition, the number of EB states is controlled by the difference in RTO and $\text{RTO}_{\max}$, so $n$ also increases with $\text{abs}(\text{RTO} - \text{RTO}_{\max})$. Therefore, if memory and processing power are in short supply, the Markov model may prove too costly to employ in real-time.

Alternatively, the renewal model has minimal computational requirements, making it advantageous when processing power and memory are limited. Still, the renewal model is less accurate; making results questionable. On the contrary, we believe a more favourable result is possible if $L$ (i.e., the number of packets lost at the end of a CA period), had better characterization. Currently, $L$ is assumed to be uniformly distributed between 1 and $W$, so $E[L] = W/2$; when in fact, the expected number of losses should be expressed by

$$E[L] = \sum_{i=1}^{W} ip(1-p)^{i-1} = \frac{1 - (1 + pW)(1-p)^W}{p}. \tag{5.51}$$

Unfortunately, the exponent in Eq. (5.51) complicates the relatively straightforward solution of $W$ (see Eq. (5.26) - (5.28)). If (5.51) were used instead of $E[L] = W/2$, numerical methods might be the only way to solve for $W$; possibly increasing processor demand or memory requirements.

9. In these experiments $\omega_{\max}$ is a function of the packet size and the RBUF.

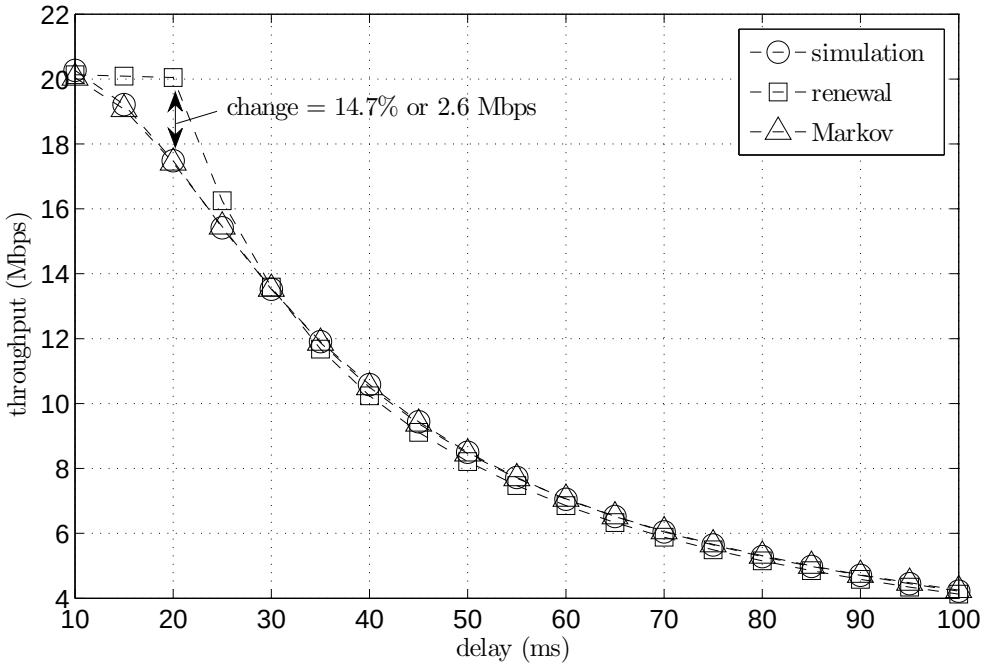


Figure 5.7: Model comparison: $p = 10^{-3}$, $b = 21$ Mbps, RBUF = 128 KB.

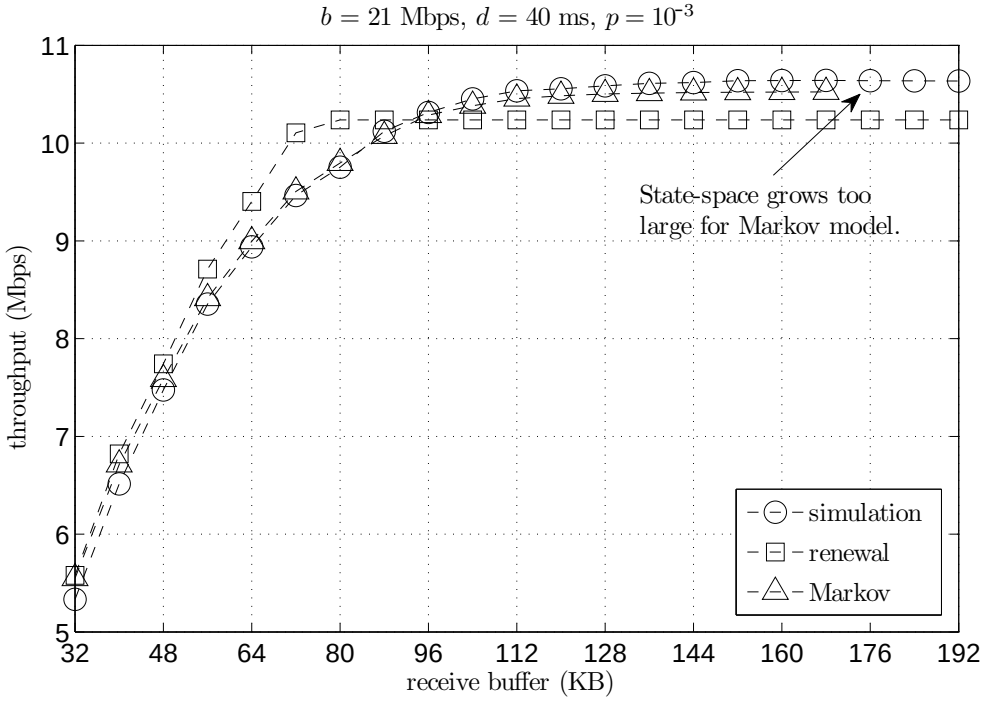


Figure 5.8: Model comparison: $p = 10^{-3}$, $b = 21$ Mbps, $d = 40$ ms.

## 5.4 Summary

In this chapter we developed a framework to model CMT. Two modelling techniques were evaluated; one based on a Markov chain, and the other using renewal theory. Although either model makes a number of assumptions regarding the mechanics of CMT, approximations were shown to be consistent with simulated results. Unfortunately, the Markov model has scalability issues; increasing its state-space with more destination addresses will undoubtedly degrade computational feasibility. On the other hand, even though renewal theory is cost effective, approximations are not always accurate.

# Chapter 6

# CMT: Congestion Window Management

As we saw in Chapter 4, the size of one destination's CWND can affect another's utilization – leading to a decrease in overall throughput. Since our new scheduling algorithm will always send to the destination with the lowest RTT and open CWND first, a single destination can end up monopolizing session resources. In this chapter we introduce a new component for CMT called congestion window management. The new manager has two primary responsibilities: (1) limit a destination's CWND to the bandwidth potential of its corresponding network path, and (2) configure the size of each CWND to maximize aggregated throughput.

The rest of the chapter is organized as follows: Section 6.1 surveys previous works related to the concept of congestion window management; Section 6.2 shows how to limit a destination's CWND; Section 6.3 offers two different schemes for congestion window optimization; Section 6.4 presents performance results; and Section 6.5 summarizes the chapter.

## 6.1 Related Work

Although we are introducing congestion window management as a new concept, we should still review any work that is related to its core functionalities. Therefore, this section is broken into two parts: while the first addresses past congestion window update polices, the second looks at optimizing buffer sizes.

### 6.1.1 Alternative Congestion Window Update Policies

An extensive survey presenting twenty years worth of TCP implementations can be found in [81]. The survey, moreover, explores a number of alternative congestion window update policies aimed at specific network problems, not uncommon to SCTP, nor CMT. Unfortunately, very few techniques from [81] suggest a solution to our particular CMT problem; i.e., how to prevent one destination address from monopolizing a shared RBUF?

For the most part, techniques for updating the CWND try to send as much information, as fast as possible, while still minimizing loss at the bottleneck link. For example, works from [82–84] approach the optimal CWND[1] in a related manner, i.e., by rapidly increasing the CWND (similar to slow-start) until noticing a loss, then applying some algorithm so that new growth approximates a horizontal asymptote.

Unlike other research from this area, the work in [85] points out an interesting relationship between achievable rate and the size of the CWND. For instance, if the network is not congested, throughput will continue to rise, but flattens when congestion sets in. Taking advantage of this notion, the authors developed a scheme called persistent noncongestion detection (PNCD) to probe for additional bandwidth. PNCD calculates a *congestion boundary*, i.e., an estimate on network limitation, to benchmark throughput potential. PNCD will then either increase or decrease a counter, depending on whether sampled throughput is above or below this congestion boundary. Finally, every time the counter equals the CWND, PNCD assumes more bandwidth is available and increases the size of the current CWND.

1. In this case, the optimal CWND is a maximum value that does not overflow the bottleneck.

### 6.1.2 Buffer Size Optimization

Up until now, all work related to buffer optimization has focused on auto-tuning the RBUF at the transport layer. When the end-to-end path has a high BDP, a large volume of data should always be in transit. Unfortunately, if the RBUF is smaller than the BDP, a reliable transport protocol like TCP or SCTP will never achieve maximum throughput. Therefore, when high speed applications like GridFTP [86] need to move large amounts of data through a network at gigabit speeds, default TCP settings are unacceptable. As a resort, the research community has developed a number of techniques to auto-size socket buffers when a high speed network is available [87–90]. Nevertheless, the most straightforward auto-tuning technique simply measures delay and available bandwidth, then sets the RBUF to the corresponding BDP.

We should mention, however, in our work we assume the RBUF to be fixed because we want to consider devices like smartphones that are limited in computing resources. Furthermore, we do not consider gigabit speeds since 4G communications are not expected to exceed 100 Mbps [91].

## 6.2 Congestion Avoidance for CMT

In this section we present a method to limit the size of a CWND, but first we motivate the work by showing how unbounded CWNDs are wasteful and only increase RTTs.

### 6.2.1 Motivation for Better Congestion Control

Even though packet scheduling can mitigate receive buffer blocking, it can also create new inefficiencies. For example, ODS (see Chapter 4) looks to send to the destination with the lowest RTT upon every transmission opportunity, and only when the

destination with the lowest RTT exhausts its CWND availability will packets be sent to an alternate destination. Unfortunately, increasing a destination's CWND using SCTP's standard congestion window update policy (which will now be referred to as $policy_1$), can inadvertently create race conditions for the RWND. When a destination address consistently offers the lowest RTT, the rate of transmission grows with CWND; gradually monopolizing the RWND along the way. Although this approach might seem optimal, as the entire RWND is always utilized, aggregated throughput can no longer be achieved. In fact, even a singlehomed association employing $policy_1$ can create unnecessary congestion and longer RTTs without increasing throughput.

During congestion avoidance (i.e., CWND $\geq$ SSTHRSH), the CWND is increased by one MTU (approximately) every RTT. At some point, however, increasing the CWND any further will only degrade network efficiency. Assume the speed of a network path is limited by only the transmission rate of its bottleneck link, so that propagation delay can be ignored. Furthermore, surmise it to say the return path is infinitely faster than the forward path so that acknowledgements experience no delay at all. Then, only one packet needs to be in flight, or in other words, the CWND only needs to be one packet long to achieve maximum throughput (i.e., the speed of the bottleneck link). On the other hand, if the propagation delay is significant (e.g., the bottleneck link can transmit the entire packet before any bits arrive at the receiver), the CWND needs to be increased before maximum throughput is realized.

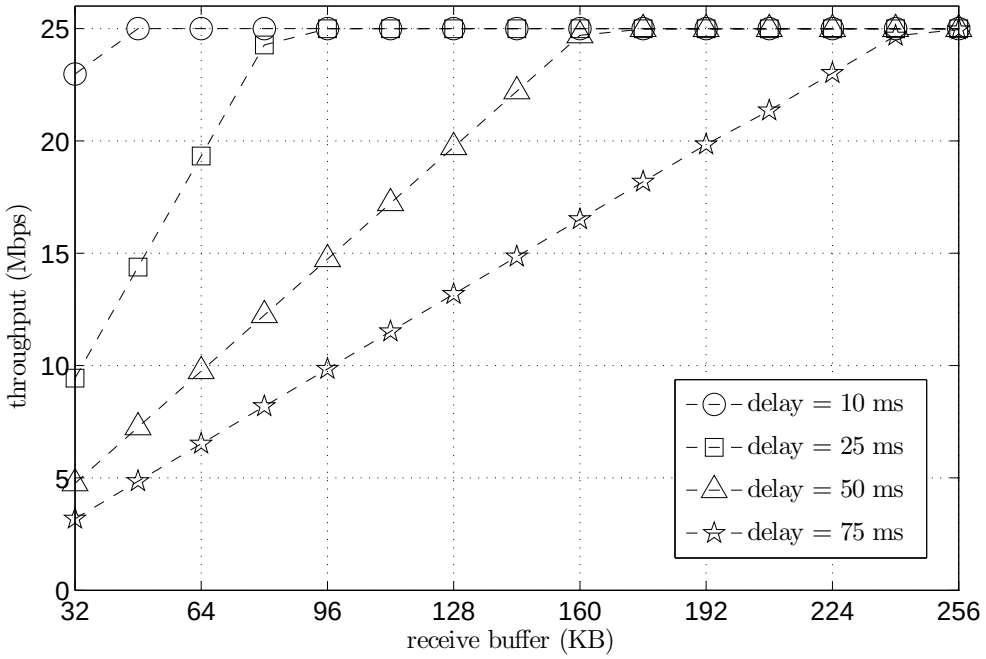


Figure 6.1: Throughput for a singlehomed connection using SCTP's standard congestion window update policy.

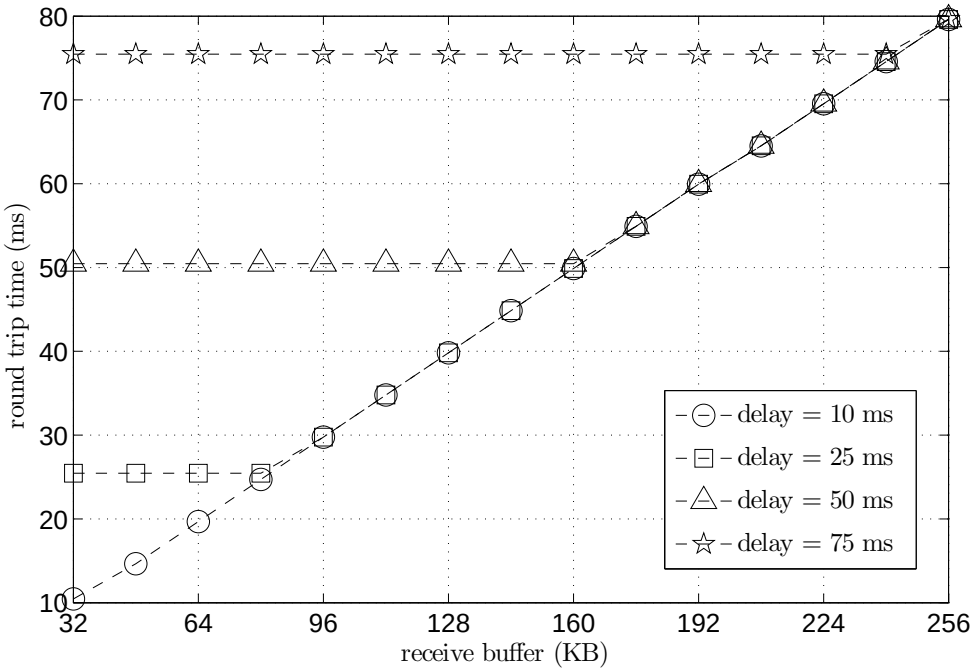


Figure 6.2: Average round trip times for a singlehomed connection using SCTP's standard congestion window update policy.

Fig. 6.1 shows how throughput is affected when packets (of size 1500 bytes) are continuously transmitted over a link with a bandwidth of 25 Mbps, while varying propagation delay and the size of the RBUF. Notice how throughput stops increasing once the RBUF reaches some threshold. Fig. 6.2, moreover, shows how RTTs continue to increase when the RBUF is pushed past this limit. What this means is that more and more packets are queueing at the bottleneck link, but the rate at which they are processed remains unchanged. When the queue finally reaches capacity, packets are dropped; even in the absence of background traffic.

### 6.2.2 Limiting Congestion Windows using Bandwidth Delay Products

To address the issues presented in Sections 4.3.5 and 6.2.1, we modified SCTP's current CWND update policy in three ways. First, we added a rule to limit each destination's CWND to the size of its corresponding BDP, so that no destination can use more resources than it requires. Next, to maintain flow control in a multihomed environment, we created a rule to limit the sum of all CWNDs to the size of the shared RBUF. Finally, a local optimization technique was applied to allow those destinations with higher bandwidth potential to grow their CWNDs before others (e.g., those with lower bandwidth potential). From now on we will refer to this new CWND update policy as $policy_2$.

Before using this method, however, one question needs to be answered: how often should bandwidth be measured? Due to background traffic, packet delays will fluctuate, changing bandwidth estimates on a per packet basis. In [90], the authors argue that bandwidth should be measured every two RTTs. For consistency, we will use the same method in this work as well.

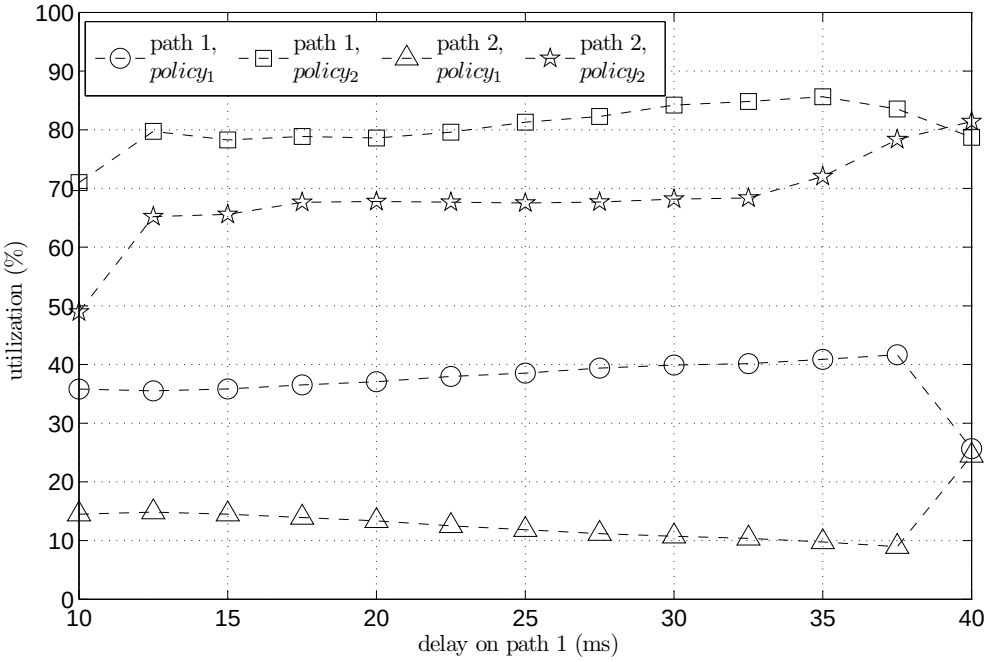


Figure 6.3: Delay-based disparity revisited: utilization.

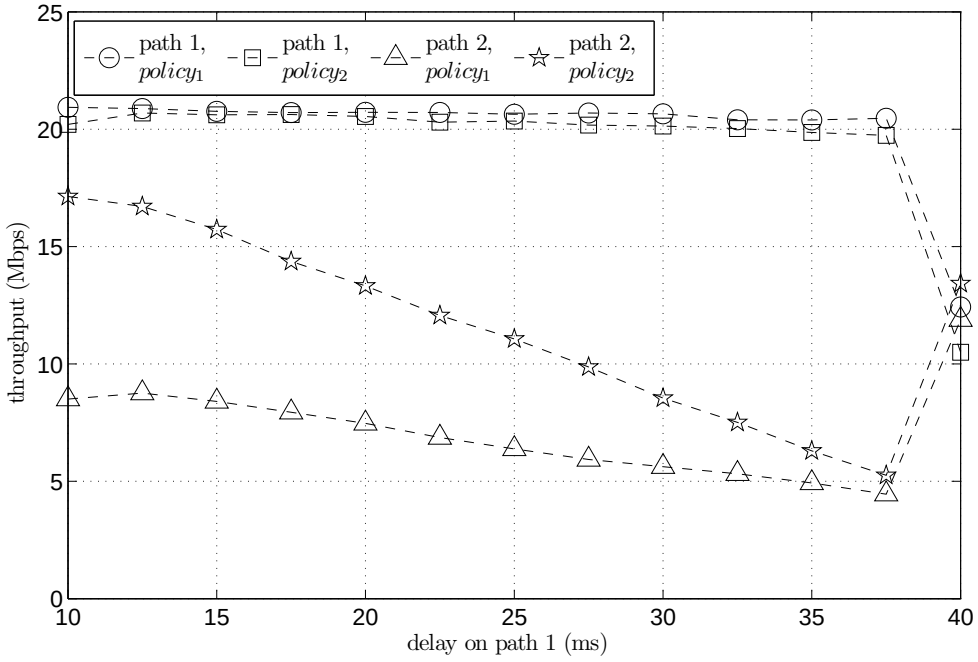


Figure 6.4: Delay-based disparity revisited: throughput.

Another problem can occur when there is a sudden shift in delay or bandwidth. Such a shift is usually caused by an increase or decrease to background traffic. Presumably, the CWND should be lowered when BDP goes down and vice versa when it goes up. Unfortunately, a large drop to BDP could stall transmission for an extended period of time while the number of outstanding packets reduce to the size of the new CWND. During this period, moreover, the current bandwidth estimate should be voided because packets need to be transmitted in order to justify a measurement. Therefore, when the CWND is greater than BDP, we only reduce the CWND every RTT so that packet transmissions are uninterrupted and bandwidth estimates are more reliable. The pseudo code for our new congestion window update policy (i.e, $policy_2$) can be found in Appendix B.

### 6.2.3 Return to Delay-based Disparity and the Utilization Problem

We will now return to some of our results from Chapter 4 to see the effect of $policy_2$ on CMT. First, we revisit Section 4.3.5 in Fig. 6.3. Here, we can see a significant improvement to path utilization. On the other hand, when we compare throughput in Fig. 6.4, we only see improved results on path 2, but aggregated throughput (i.e., the sum of both paths) has grown. Although $policy_1$ still allows one path to flourish, by ignoring CWND limits it can also prevent other paths from capitalizing on additional performance gains. To the contrary, $policy_2$ realizes these additional performance gains by using resources more efficiently.

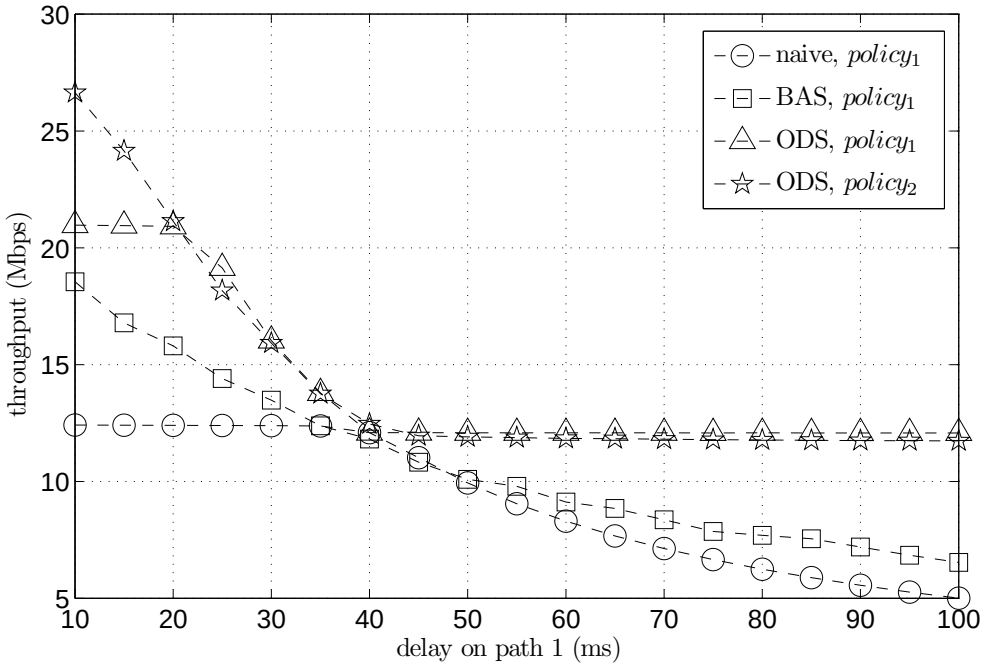


Figure 6.5: Delay-based disparity revisited: throughput results when RBUF is 64 KB.

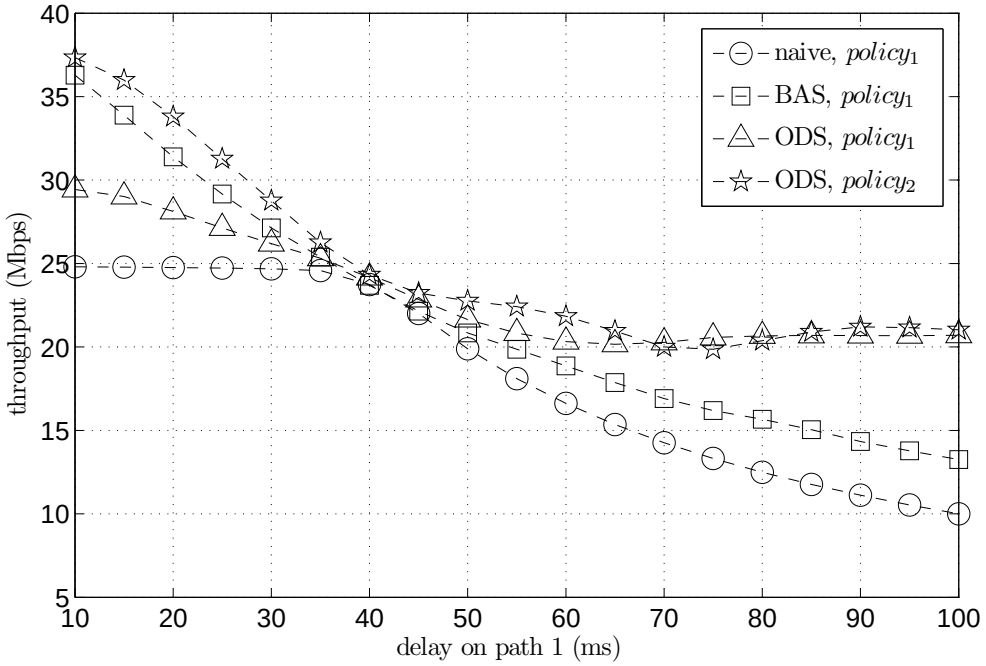


Figure 6.6: Delay-based disparity revisited: throughput results when RBUF is 128 KB.

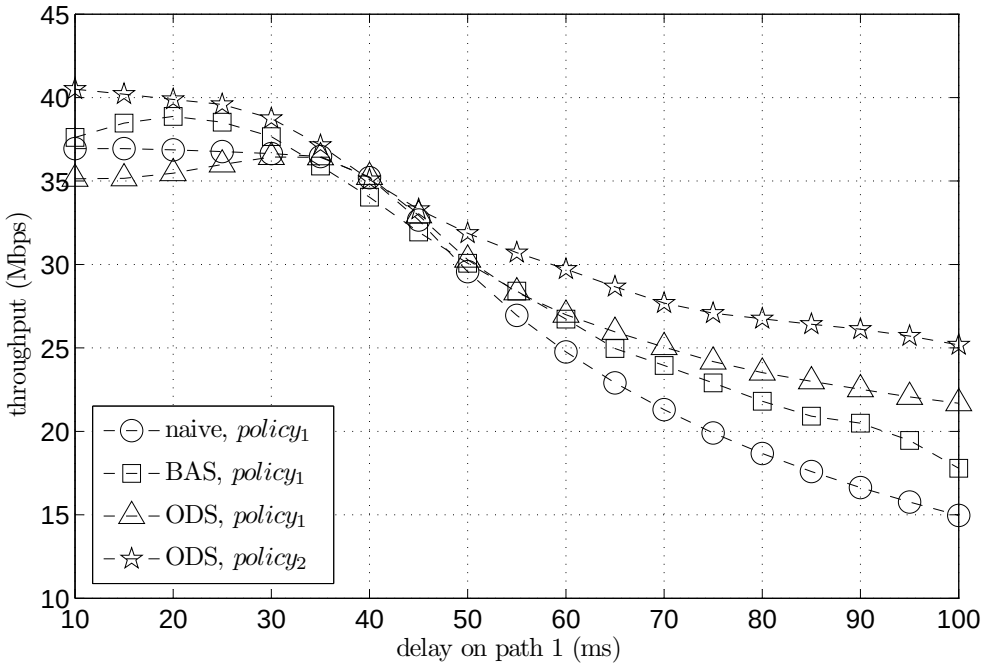


Figure 6.7: Delay-based disparity revisited: throughput results when RBUF is 192 KB.

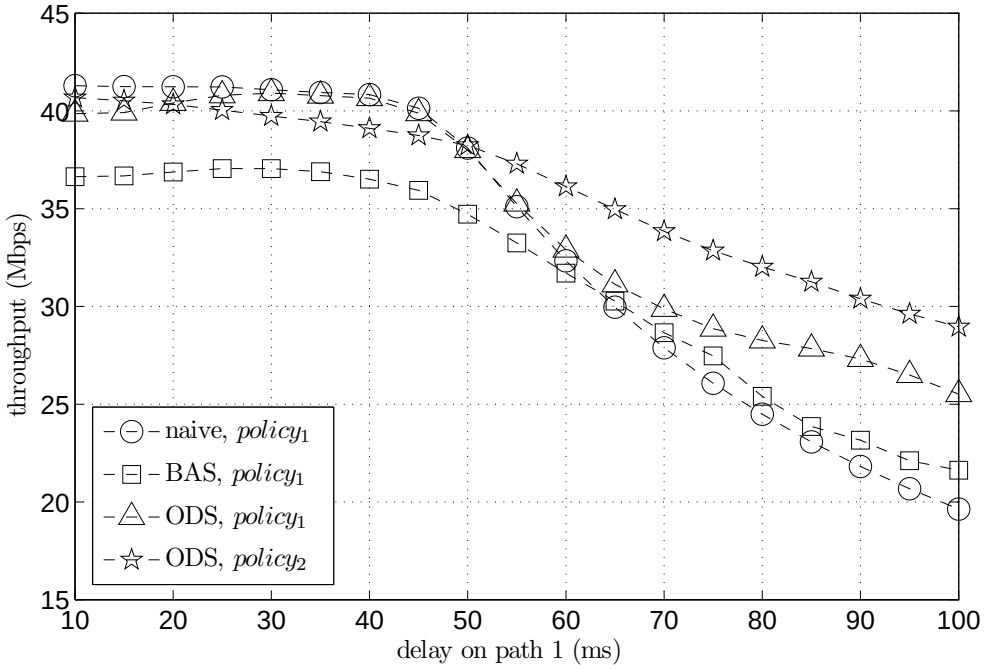


Figure 6.8: Delay-based disparity revisited: throughput results when RBUF is 256 KB.

Next, we will compare $policy_2$ with the results from Section 4.3.2 in Figs 6.5 - 6.8. Notice when $policy_2$ is employed ODS has a much stronger advantage over the other two scheduling approaches[2].

In the areas where $policy_2$ does not have an advantage (e.g., when delay is less than 50 ms in Fig. 6.8), the difference is only marginal. The main reason why $policy_2$ is not always advantageous is actually because it limits CWNDs while $policy_1$ does not. For example, following the loss of a packet that was sent to a particular destination, both policies would cut that particular destination's CWND in half. Since $policy_2$ limits CWNDs, after every loss a certain amount of time is required for a destination's CWND to grow back to its maximum operating point. Moreover, during this growth period throughput is always less than optimal. Conversely, $policy_1$ does not limit CWNDs, allowing them to grow so high that cutting a CWND in half has little impact on throughput.

## 6.3 Congestion Window Optimization for CMT

In this section, we present two ways of optimizing CMT; while one method is greedy, taking only instantaneous throughput into account, the other uses average throughput to place limits on the CWND of each destination address.

### 6.3.1 Dynamic Optimization

Our first method tries to maximize throughput in a greedy fashion by adjusting CWNDs based on potential throughput; from here on out we will refer to this method

2. Since $policy_2$ showed no improvement when paired with the other two schedulers, we chose not to include their results, simply for presentation purposes.

as *dynamic* congestion window management. In fact, we have already used dynamic congestion window management in $policy_2$.

Dynamic congestion window management seeks to improve instantaneous throughput by simultaneously lowering one CWND while raising another when the sum of CWNDs is equal to the size of the RBUF. For example, if the sum of CWNDs is equal to the size of the RBUF, but the destination with the highest available bandwidth can improve throughput by increasing its CWND; intuitively, another destination address will first have to lower its CWND before another can be raised. This is accomplished by ranking each destination. A destination with a higher rank can reduce the CWND of a destination with a lower rank. Moreover, destination rank is decided by available bandwidth, with delay breaking a tie. For example, if two destination addresses have the same bandwidth, they are differentiated by their delays. This method is particularly useful when the RBUF is small and all destinations have a low probability of packet loss.

### 6.3.2 Static Optimization

What if a higher ranked destination also had a higher loss rate, so its CWND grew rapidly but was also cut back on a more frequent basis? In that case, lowering one CWND to raise another would still improve instantaneous throughput, but over the long run, average throughput could be suffering. To address this issue, we suggest a more static approach to congestion window management where a limit is placed on the CWND of a destination address. In this way ranks are eliminated, and an optimization technique is used to find the CWND limit of each destination address that will maximize throughput. In this subsection, we present an integer linear program (ILP) that calls either of the models from Chapter 5 to search for the best configuration of

CWND limits; we call this approach *static* congestion window management.

Additionally, even though all CWNDs will have a limit, they are not always bound to this limit. For example, a destination can grow its CWND above its limit if the following two conditions are true: (1) the sum of the CWNDs is less than the RBUF, and (2) the CWND is less than its BDP. Limits are still the rule, however, so if another destination tries to increase its CWND and finds the sum of CWNDs equal to the RBUF, a CWND operating above its limit will be reduced.

An ILP will now be formulated to maximize the throughput of CMT. First, we let the function $\eta(b_i, d_i, p_i, t_i^{\max}, c_i^{\max})$ return the expected throughput[3] of a single path transfer; provided that $i \in N = \{1, \ldots, n\}$ and $b_i$, $d_i$, $p_i$, $t_i^{\max}$, and $c_i^{\max}$ are respectively the bandwidth, delay, loss rate, $\text{RTO}^{\max}$, and CWND limit of destination $i$. Assuming, furthermore, the size of the RBUF is given by $r$, our ILP for static congestion window management is expressed in just five equations:

$$\textbf{maximize} \sum_{i}^{n} \eta(b_i, d_i, p_i, c_i^{\max}, t_i^{\max}) \tag{6.1}$$

$$\textbf{s.t.} \sum_{i}^{n} c_i^{\max} \leq r, \tag{6.2}$$

$$1 \leq c_i^{\max} \leq \lceil b_i \cdot d_i + 1 \rceil, \tag{6.3}$$

$$c_i^{\max} \in \mathbb{Z}, \tag{6.4}$$

$$\forall i \in N. \tag{6.5}$$

### 6.3.3 Heuristic

Using brute force, our ILP can only be solved in exponential time. For example, assuming $\lceil b_i \cdot d_i + 1 \rceil \geq r, \forall i \in N$, then solving the static congestion window man-

3. Approximated using either the Markov or renewal model (see 5.2.3 and 5.2.4).

agement problem will have a computational complexity of $\mathcal{O}(r^n)$. Alternatively, the problem could be solved in polynomial time if we assumed $n$ to be constant (e.g., $n = 2$ for a smartphone with 802.11 and GSM/UMTS interfaces). Even still, if $r$ is large enough, a real-time system could find an exhaustive search intolerable.

In a real-time system, deadlines are critical; so process runtimes are always deterministic. If it so happens that the runtime to solve our ILP is intolerable (i.e., the real-time system cannot wait for a solution), finding an optimal solution may not be possible. As a compromise, however, a heuristic can usually find one or two satisfactory solutions – albeit suboptimal – in a reasonable amount of time. Next, we will describe a simple heuristic to solve the static congestion window management problem when finding an optimal solution proves too costly.

Our heuristic simply reduces the number of searches needed to find a solution by using a subset of values available for $c_i^{\max}$. For example, if $n = 2$, $r = 100$ packets, $b_1 = b_2 = 1000$ p/s, and $d_1 = d_2 = 100$ ms, both $c_1^{\max}$ and $c_2^{\max}$ could take on any value in the set $\mathbb{C} = \{x : x \in \mathbb{Z} : 1 \leq x \leq r\}$. An exhaustive search, therefore, would need a total of 10000 iterations to generate an optimal solution. Alternatively, if we used the set $\{2x : x \in \mathbb{C}, x \leq \frac{|\mathbb{C}|}{2}\}$, the total number of iterations drops to 2500. Moreover, if we used the set $\{10x : x \in \mathbb{C}, x \leq \frac{|\mathbb{C}|}{10}\}$, just 100 iterations are needed. A general formula for our heuristic (i.e., a subset of CWND limits for destination $i$), is given by

$$\{kx : 1 \leq x \leq \left\lfloor \frac{a}{k} \right\rfloor, a = \min(r, \lceil b_i \cdot d_i + 1 \rceil), k \leq a\}. \tag{6.6}$$

## 6.4 Performance Results

Using the network topology from Fig. 4.2, we will now study the impact of static and dynamic congestion window management on the performance of CMT. For easy

comparison, we simply calculated the percent gain, in terms of throughput, when using the static approach instead of the dynamic one. For brevity, moreover, we only implemented static congestion window management using the Markov model[4].

### 6.4.1 Simulation Parameters

In all tests the probability of packet loss to destination 1 (i.e., $p_1$) was varied between $10^{-4}$ and $10^{-1}$, with the following four parameters remaining constant: 1) $b_1 = 21$ Mbps, 2) $d_1 = 40$ ms, 3) $t_1^{\max} = t_2^{\max} = 60$ seconds, and 4) packet size = 1500 bytes. Furthermore, to create a variety of scenarios, in each test we changed one of the following: $r, p_2, b_2$, and $d_2$. Unless otherwise specified, Table 6.1 lists all other simulation parameters[5].

### 6.4.2 Results and Discussion

The results of our findings are shown in Figs. 6.9 - 6.12. In most cases the static approach improved performance, even as high as 12%. Unfortunately, there were some instances where no improvement could be made, and some cases where dynamic congestion window management yielded even better results.

In the case of Fig. 6.9, the RBUF is 128 KB and destination 1 will use up to 108 KB; leaving just 20 KB available for destination 2. When $b_2$ is low, destination 2 does not need a very high CWND to maximize throughput (e.g., at 5 Mbps, $c_2^{\max} = 26$ KB), so the static method has little impact. But when $b_2$ is larger, destination 2 will need a higher CWND, and thus more of the RBUF to leverage performance. Furthermore,

4. Since the Markov model can support both independent and correlated loss models, we now assume all losses to be independent and $p$ to be the probability of packet loss.

5. Parameter choices followed the same reasoning found in Section 5.3.

| | | |
|---|---|---|
| application layer | continuous file transfer | 60 minutes |
| SCTP end-points | packet size | 1500 bytes |
| | SBUF | $\infty$ |
| | RBUF | 64, 96, 128, 192, 256 KB |
| | $\text{RTO}_{\text{max}}$ | 60 seconds |
| | RTO parameters | $\alpha = 0.125, \beta = 0.25$ |
| network path 1 | bandwidth | 21 Mbps |
| | propagation delay | 40 ms |
| | prob. packet loss | $10^{-4}$ - $10^{-1}$ |
| network path 2 | bandwidth | 5, 10, 15, 20 Mbps |
| | propagation delay | 30, 40, 50, 60 ms |
| | prob. packet loss | 0,$10^{-4}$,$10^{-3}$,$10^{-2}$,$10^{-1}$ |

Table 6.1: Static vs. Dynamic CWND Management: simulation parameters.

the static approach is always more effective when $p_1 \in [10^{-3}, 10^{-2}]$ because it makes sense to increase the CWND of a stable (yet lower ranked) destination when a higher rank becomes volatile.

A similar explanation can be made for Fig. 6.10. Again, destination 1 is ranked highest (i.e., $b_1 > b_2$); so when the RBUF is lower than $c_1^{\max}$ and $p_1$ is minimal, destination 2 will send very little. In any event, both destinations have comparable bandwidth, so the difference in choosing one over the other is marginal. As $p_1$ increases, however, the dynamic scheme will let destination 2 grow its CWND, but only temporarily. Every time destination 1 lowers its CWND (e.g., due to packet loss) destination 2's CWND can grow again, but it is for this reason that the static method is better; raising its CWND only to lower it again at the behest of another – more unstable destination – is an inefficient use of resources. Of course, static congestion window management becomes less significant once the RBUF becomes large enough to support both destinations' bandwidth potentials, regardless of loss rate.

In Fig. 6.11, improvement goes to zero as loss rate increases on path 2. The main reason, however, does not change; as loss rate increases, average CWND reduces

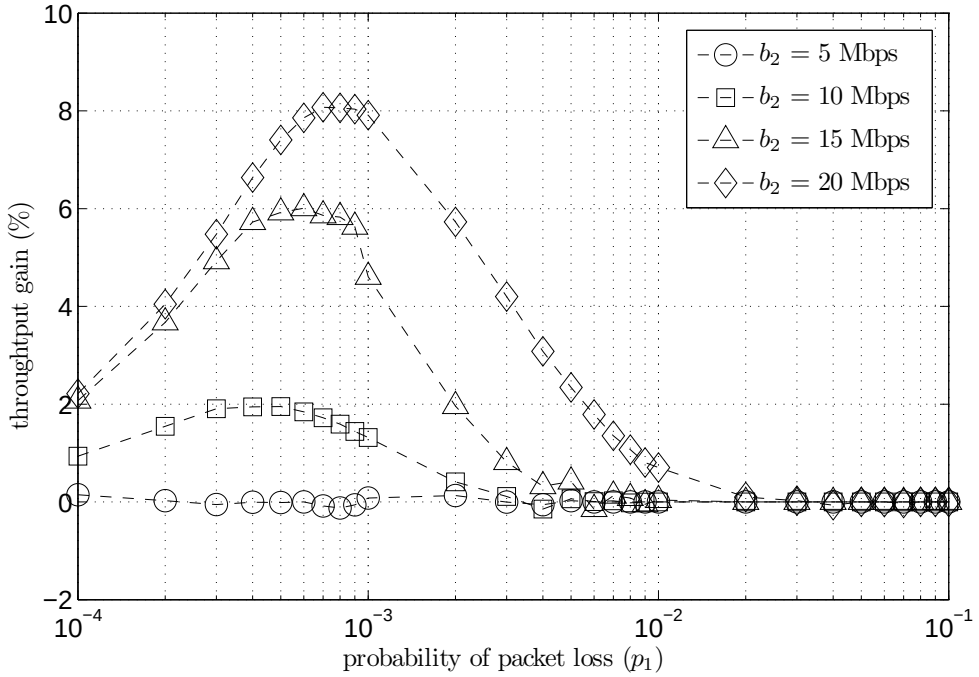


Figure 6.9: Static vs. Dynamic: $d_2 = 40$ ms, $p_2 = 10^{-4}$, $r = 128$ KB.

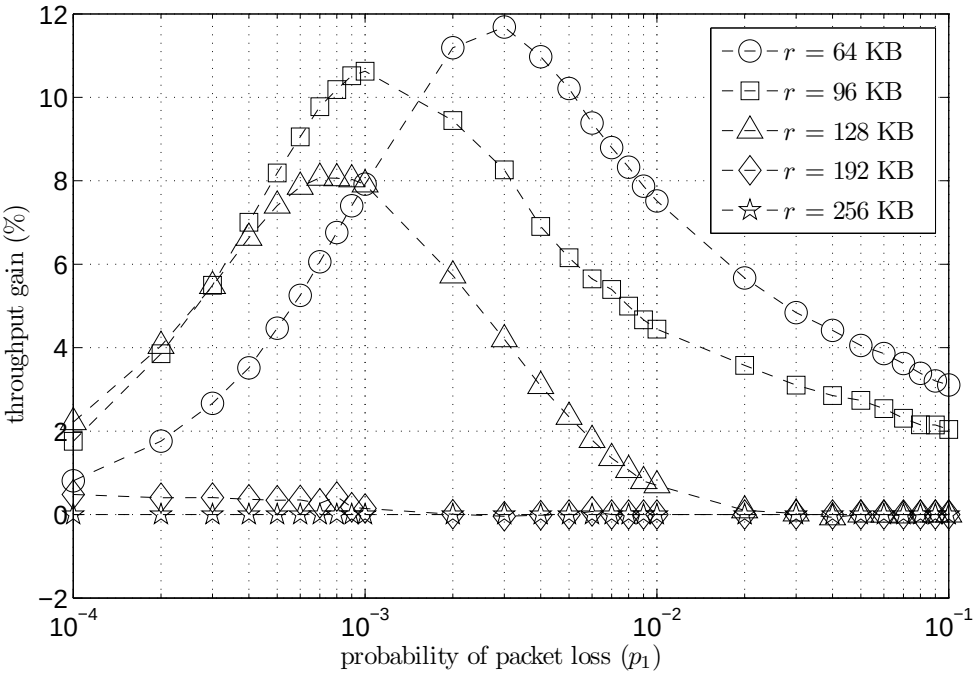


Figure 6.10: Static vs. Dynamic: $b_2 = 21$ Mbps, $d_2 = 40$ ms, $p_2 = 10^{-4}$.

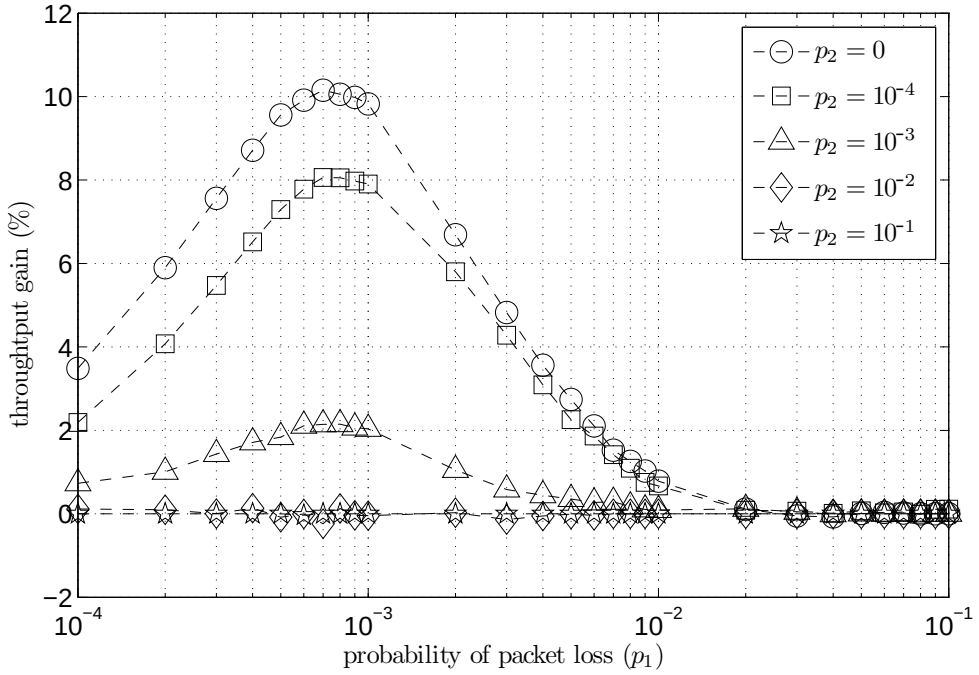


Figure 6.11: Static vs. Dynamic: $b_2 = 20$ Mbps, $d_2 = 40$ ms, $r = 128$ KB.

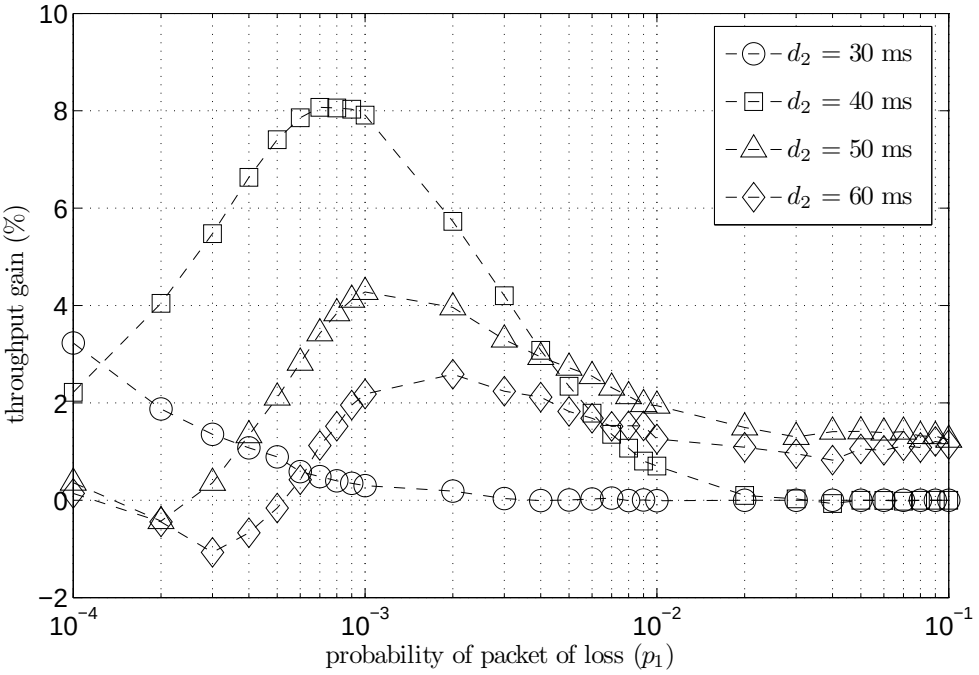


Figure 6.12: Static vs. Dynamic: $b_2 = 20$ Mbps, $p_2 = 10^{-4}$, $r = 128$ KB.

and less of the RBUF is needed by a destination. Therefore, when the RBUF is large enough, or even underutilized, static management can do no better.

Contrary to what we have seen so far, Fig. 6.12 shows us that static management is not always the better choice. For the majority of the plot, however, static management is dominant; boosting performance as much as 8%. But when the delay on path 2 is considerably large (compared to path 1), there are areas where dynamic management is more desirable (e.g., $p_1 < 10^{-3}$). Clearly, path 1 is most volatile when $p_1 \in [10^{-3}, 10^{-2}]$, and in Figs. 6.9 - 6.11, static management was able to capitalize. So why then, is this not the case when there is a disparity in path delays?

To understand this phenomenon, we first have to realize that static management does not offer an optimal solution for the dynamic method because the ILP assumes that CWND limits will prevent CWND growth beyond a certain point. Even after we apply static management, however, we still allow some CWNDs to go beyond their limits every so often. Moreover, destinations with lower RTTs will benefit the most when they push their CWNDs above the limit because of faster growth rates; growing a CWND faster means more bandwidth is consumed in a shorter period of time. Unfortunately, this growth is only temporary and the ILP does not account for any short term gains.

We will now explain how RTTs can affect short term gains through using a simple example. Given the network in Fig. 3.2, we assume $n = 2$, RBUF = 48 KB, packets = 1000 KB, and $d_1 = 50$ ms. Furthermore, we assume destination 1 to have the highest bandwidth potential so that it is favoured over destination 2 when the sum of CWNDs is equal to the RBUF. Assuming the CWNDs of destination 1 and 2 are initially 32 and 16 KB, respectively; then at time 100 ms destination 1 invokes a fast recovery and drops its CWND by half. Fig. 6.13 now illustrates what will happen to destination 2 given a RTT of 50 or 100 ms. Clearly, when RTT is 50

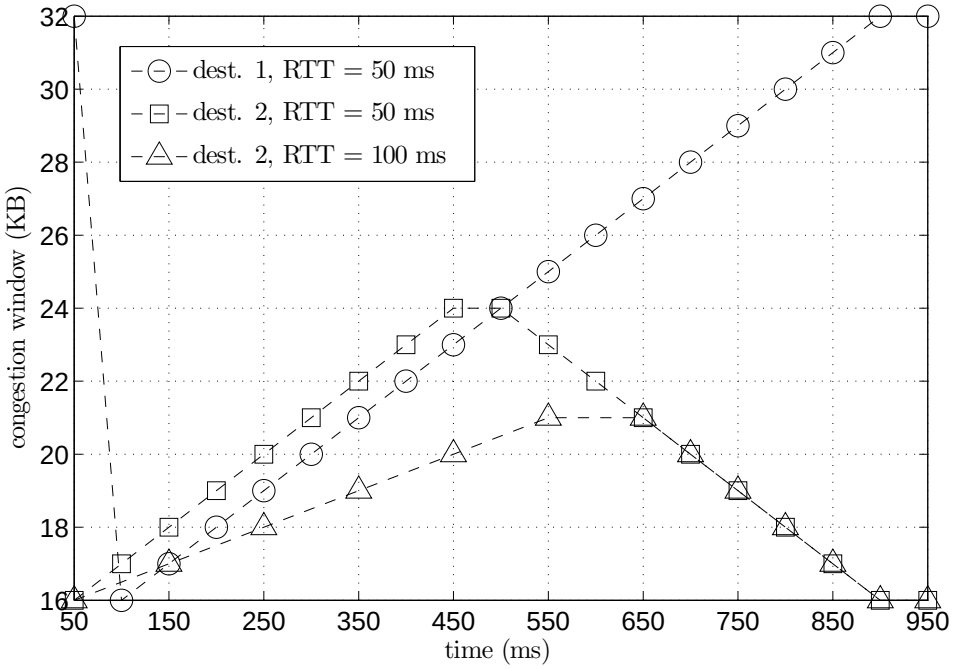


Figure 6.13: Short term gains during dynamic CWND management.

ms, destination 2 can utilize more resources during the time it takes destination 1 to recover. Using numerical integration and calculating the average sum of CWNDs over the interval gives us 20.9 KB when RTT is 50 ms, but only 18.9 KB when RTT is 100 ms. Unfortunately, since we do not have a model that encompasses these short term gains, the static method cannot always guarantee improved results.

### 6.4.3 Heuristic Results

Before leaving this section we offer some results using our simple heuristic. In this experiment, we varied $b_1$, but kept the following parameters constant: $r = 128$ KB, $b_2 = 10$ Mbps, $d_1 = 25$ ms, $d_2 = 50$ ms, $t_1^{\max} = t_2^{\max} = 60$ seconds, and packet size = 1500 bytes. Moreover, we looked at how the heuristic faired with increasing values of $k$. Figs. 6.14 and 6.15 display the results of this test.

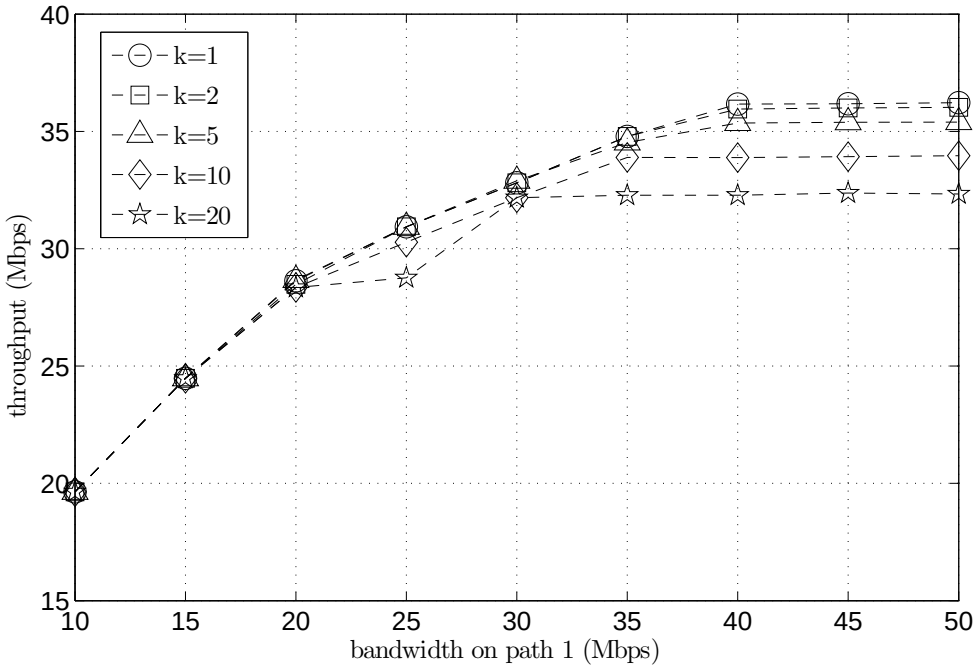


Figure 6.14: Heuristic evaluation: throughput.

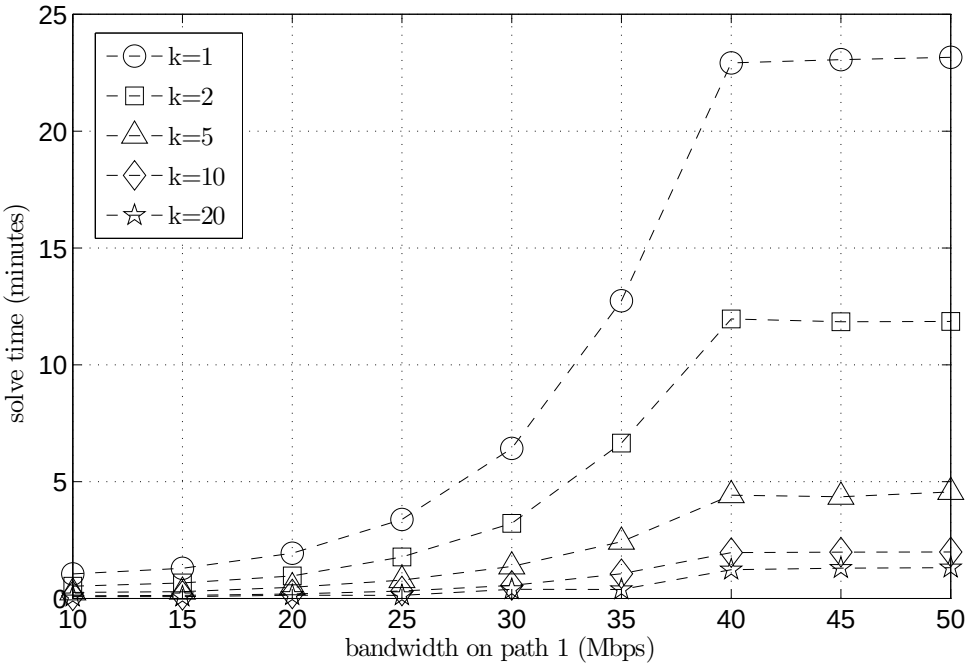


Figure 6.15: Heuristic evaluation: solve time.

Not surprisingly, the size of $k$ makes little difference in terms of throughput when $b_1$ is small. This is because the scope of the problem (or the search space) is already quite low, and lowering it any any further will have marginal impact. Nevertheless, increasing bandwidth will increase BDP which creates more CWND limits to choose from. Therefore, using a smaller value of $k$ ensures a more refined choice during static congestion window management, and leads to higher throughput values as is shown in Fig. 6.14.

Unfortunately, a smaller $k$ also means more iterations which takes more time to find an optimal solution. Finally, Fig. 6.15 shows how lowering the value of $k$ can save a considerable amount of time to solve the static congestion window management problem.

## 6.5 Summary

A new concept called congestion window management was presented in the chapter. When CMT is employed, the size of one destination's CWND can affect the utilization of another. Moreover, unless CWNDs are carefully managed, performance can suffer. Our approach divides congestion window management into two subproblems: (1) prevent a CWND from monopolizing session resources; (2) configure the size of each CWND in order to maximize throughput. For problem (1), a new CWND update policy was developed. The new update policy sets a limit on a destination's CWND by monitoring bandwidth and delay so that CWND $\leq$ BDP. Problem (2), however, was solved in two different ways: the first used a greedy optimizer that dynamically adjusts the size of CWNDs based on instantaneous throughput, and the second applied the mathematical models from Chapter 5 to generate a set of CWND limits that would maximize average throughput.

# Chapter 7

# Conclusion

In this thesis, we explored a number of topics surrounding concurrent multipath transfer (CMT) and multihomed devices. Our goal was to further the research of these areas in hopes that someday CMT would become a standard part of *smartphone* technology. Even though that day is still to come, we feel that our contributions to CMT will have a positive impact on any future research in the area of transport layer multihoming. We will now summarize the work presented in the thesis and give our recommendation for extending the research.

## 7.1 Thesis Summary

In Chapter 2, we reviewed SCTP and investigated state-of-the-art multihoming techniques. SCTP is a successor to TCP. Unlike TCP, however, SCTP allows an Internet application to exploit additional network resources found in multihomed devices, such as smartphones. A thorough review of SCTP along with the techniques used for transport layer multihoming gave way to three main areas of study: i.e., handover management, concurrent multipath transfer, and cross-layer activities. Even though a number of strategies have been applied to these areas, many open problems still remain.

In Chapter 3, we developed a transport layer architecture and a network topology to study CMT in a multihomed system. The new transport layer architecture

adds a scheduler and a destination manager to the sender's design. Furthermore, the network topology offers a general framework for multihomed systems; such that any number of paths may exist between sender and receiver, where each path is defined by a set of independent performance characteristics (e.g., bandwidth, delay, and loss rate).

In Chapter 4, we investigated the receive buffer blocking problem associated with CMT. Receive buffer blocking is caused by out-of-order arrivals at the receiver. Our approach to the problem was intelligent packet scheduling; specifically, we developed an algorithm called the on-demand scheduler (ODS) for CMT under delay-based disparity. Unlike its predecessors, ODS waits for a transmission opportunity before assigning packets to a destination address. When congestion and flow control allow transmission, ODS searches the SBUF in a recursive manner, looking for a packet that cannot be delivered to another destination sooner. For the most part, ODS outperformed the previous scheduling algorithms under delay and bandwidth-based disparity. Unfortunately, ODS could not over come loss-based disparity. Unless losses are predictable, it seems any scheduler for CMT will always succumb to receive buffer blocking when packets are lost and need to be retransmitted.

Assuming an ideal scheduler (i.e., one that can predict losses), a framework for modelling CMT was developed in Chapter 5. Two separate models were created; one based on a Markov chain, and the other using renewal theory. The Markov model had the highest accuracy but does not scale well; state-space increases dramatically when more destination addresses are added to the problem domain. To the contrary renewal theory was not as accurate but more cost effective. If a real-time application needed to approximate the throughput of CMT, renewal theory would most likely be the only candidate as it can generate results with minimal computing effort.

Finally, Chapter 6 introduced a new concept for CMT called congestion window

management. Since SCTP's standard congestion window update policy can restrict the performance capabilities of ODS, a new update policy was created to limit the size of CWNDs; specifically to the BDP of corresponding network paths. When compared to SCTP's standard congestion window update policy, our new policy increases destination utilization and improves aggregated throughput.

Also addressed in Chapter 6 was congestion window optimization for CMT. Even if each CWND is limited to the BDP of its corresponding path, the sum of all CWNDs must be less than the size of the RBUF; therefore one destination address can still prevent another from growing its CWND. Depending on bandwidth, moreover, the size of one CWND can have a dramatic effect on throughput. To address this issue we developed two strategies: the first used a greedy optimizer that dynamically adjusts the size of CWNDs based on instantaneous throughput, while the second used the mathematical models from Chapter 5 in an ILP to generate a set of CWND limits that maximize average throughput. Typically, the second strategy was better, or at least as good as the first. But the second strategy does not consider short term gains that can occur after loss events; so in some cases the greedy strategy was the better choice, even without global optimization. Furthermore, the static approach needs to search through a set of CWND limits, for each destination address, in order to determine optimality. Unfortunately, the search will not always meet real-time constraints. Therefore, we also developed a heuristic to a find feasible solution for the congestion window management problem, but in a more reasonable frame.

## 7.2 Future Work

Although ODS can improve the performance of CMT, before every transmission, a recursive algorithm searches the SBUF to find the next packet that will minimize

reordering. Depending on the number of destination addresses as well as the size of the SBUF, the processing delay created by ODS may become apparent. Since the majority of modern desktop and laptop computers possess a surplus of processing and memory resources, we felt it reasonable to assume processing delays to be negligible when a file is being downloaded. With that said, mobile devices, like smartphones, are less abundant when it comes to computing power. A smartphone, moreover, needs to be as efficient as possible, especially since its only power source is a battery. If the process were reversed, and the mobile was to upload a file to a receiver with multiple destination addresses, ODS could be overly taxing. Future work, therefore, should focus on the processing requirements and scalability of ODS under a variety of specific computing architectures.

To scale back computational costs, we could approximate the number of packets delivered to one destination within the time it takes to send a single packet to another one. Using a bandwidth estimate and path variables (e.g., CWND, RTT), a closed-form expression could be developed for the purpose of selecting a packet from the SBUF that minimizes receive buffer blocking; although some blocking would have to be tolerated, such an approach should reduce computational complexity to relieve system resources of ODS's exhaustive scheduling process.

Another issue that still plagues CMT is receive buffer blocking under loss-based disparity. When a packet is lost, a hole is created and packets transmitted after a loss will have to wait in the RBUF for a retransmission. Alternatively, if packet loss events could be characterized using a probability distribution function, we could then try to predict when losses might occur. In anticipation of a loss, we could send redundant packets over a different path so that retransmissions would become unnecessary.

In terms of modelling, we assumed perfect scheduling to avoid receive buffer blocking caused from loss-based disparity. However, until we can generate a more ideal

scheduler, the CMT model from this thesis is only theoretical. Moreover, static congestion window management depends on accurate modelling to configure the CWND of each destination address. In case an ideal scheduler proves too complicated for implementation, future work will have to incorporate receive buffer blocking into the CMT model.

Finally, we compared two optimization techniques for congestion window management, but found that sometimes one is better than the another. If we expect to implement CMT in a real-time system, we will want our optimization techniques to have greater reliability. Unfortunately, our static method currently ignores the short term gains that occur when one destination cuts back its CWND (due to loss); resulting in an increased number of packet transmissions to another destination, albeit only temporarily. By including these short term gains into the model, we should be able to offer a better prediction on performance results. In addition, by removing some of the search space with the help of a heuristic, we were able to reduce the amount of time to find a feasible solution. Therefore, future work in this area could take advantage of more popular meta-heuristic strategies, such as simulated annealing, genetic algorithms, or tabu search. Alternatively, we could also take a completely different approach to the static congestion window management problem altogether, by formulating an optimal decision policy using a Markov decision process.

# References

[1] T. D. Wallace and A. Shami, "A review of multihoming issues using the stream control transmission protocol," *IEEE Commun. Surv. Tutorials*, no. 99, pp. 1–14, 2011. 6

[2] R. Stewart, "Rfc 4960: stream control transmission protocol," *Request for Comments, IETF*, 2007. 8, 19, 24, 56

[3] R. Stewart, M. Ramalho, Q. Xie, M. Tuexen, and P. Conrad, "Rfc 3528: Stream control transmission protocol (sctp) partial reliability extension," *Request for Comments, IETF*, 2004. 8

[4] M. Fiore, C. Casetti, and G. Galante, "Concurrent multipath communication for real-time traffic," *Comput. Commun.*, vol. 30, no. 17, pp. 3307–3320, 2007. 8, 36

[5] R. Stewart and C. Metz, "Sctp: New transport protocol for tcp/ip," *IEEE Internet Comput.*, vol. 5, no. 6, pp. 64–69, 2001. 11

[6] A. Caro, J. Iyengar, P. Amer, S. Ladha, G. Heinz, and K. Shah, "Sctp: A proposed standard for robust internet data transport," *Computer*, vol. 36, no. 11, pp. 56–63, 2003. 11

[7] S. Fu and M. Atiquzzaman, "Sctp: State of the art in research, products, and technical challenges," *IEEE Commun. Mag.*, vol. 42, no. 4, pp. 64–76, 2004. 11

[8] A. L. Caro, P. D. Amer, and R. R. Stewart, "Retransmission policies for multihomed transport protocols," *Comput. Commun.*, vol. 29, no. 10, pp. 1798–1810, 2006. 12

[9] D. Le, X. Fu, and D. Hogrefe, "A review of mobility support paradigms for the internet," *IEEE Commun. Surv. Tutorials*, vol. 8, no. 1, pp. 38–51, 2006. 13

[10] M. Atiquzzaman and A. Reaz, "Survey and classification of transport layer mobility management schemes," in *Proc. IEEE Intl. Symposium on Personal, Indoor and Mobile Radio Communications*, vol. 4, 2005, pp. 2109–2115. 13

[11] R. Stewart, Q. Xie, M. Tuexen, S. Maruyama, and M. Kozuka, "Rfc 5061: Stream control transmission protocol (sctp) dynamic address reconfiguration," *Request for Comments, IETF*, 2007. 13

[12] I. Aydin, W. Seok, and C. Shen, "Cellular sctp: a transport-layer approach to internet mobility," in *Proc. IEEE Intl. Conf. Computer Communications and Networks*, 2003, pp. 285–290. 15

[13] S. Fu, L. Ma, M. Atiquzzaman, and Y. Lee, "Architecture and performance of sigma: A seamless handover scheme for data networks," in *Proc. IEEE Intl. Conf. on Communications*, vol. 5, 2005, pp. 16–20. 15, 22

[14] A. Ezzouhairi, A. Quintero, and S. Pierre, "A new sctp mobility scheme supporting vertical handover," in *Proc. IEEE Intl. Conf. on Wireless and Mobile Computing, Networking and Communications*, 2006, pp. 205–211. 15

[15] W. Stevens, "Rfc 2001: Tcp slow start, congestion avoidance, fast retransmit, and fast recovery algorithms," *Request for Comments, IETF*, 1997. 19

[16] X. Yan, Y. Ahmet Sekercioglu, and S. Narayanan, "A survey of vertical handover decision algorithms in fourth generation heterogeneous wireless networks," *Comput. Netw.*, vol. In Press, Corrected Proof, 2010. 22

[17] A. Sgora and D. Vergados, "Handoff prioritization and decision schemes in wireless cellular networks: a survey," *IEEE Commun. Surv. Tutorials*, vol. 11, no. 4, pp. 57–77, 2009. 22

[18] J. Fitzpatrick, S. Murphy, M. Atiquzzaman, and J. Murphy, "Using cross-layer metrics to improve the performance of end-to-end handover mechanisms," *Comput. Commun.*, vol. 32, no. 15, pp. 1600–1612, 2009. 23

[19] I. Rec, "G. 107-the e model, a computational model for use in transmission planning," *Intl. Telecommunication Union*, 2003. 23

[20] L. Ma, F. Yu, and V. Leung, "Performance improvements of mobile sctp in integrated heterogeneous wireless networks," *IEEE Trans. Wireless Commun.*, vol. 6, no. 10, pp. 3567–3577, 2007. 24, 25

[21] S. Kashihara, K. Iida, H. Koga, Y. Kadobayashi, and S. Yamaguchi, "Multi-path transmission algorithm for end-to-end seamless handover across heterogeneous wireless access networks," in *Proc. Intl. Workshop on Distributed Computing*, 2003, pp. 836–836. 25

[22] S. Koh, M. Chang, and M. Lee, "msctp for soft handover in transport layer," *IEEE Commun. Lett.*, vol. 8, no. 3, pp. 189–191, 2004. 26

[23] E. Ribeiro and V. Leung, "Asymmetric path delay optimization in mobile multihomed sctp multimedia transport," in *Proc. ACM Workshop on Wireless Multimedia Networking and Performance Modeling*, 2005, pp. 70–75. 26

[24] L. Budzisz, R. Ferrús, A. Brunstrom, K. Grinnemo, R. Fracchia, G. Galante, and F. Casadevall, "Towards transport-layer mobility: evolution of sctp multihoming," *Comput. Commun.*, vol. 31, no. 5, pp. 980–998, 2008. 26, 43

[25] R. Ludwig and R. Katz, "The eifel algorithm: Making tcp robust against spurious retransmissions," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 30, no. 1, pp. 30–36, 2000. 28

[26] E. Blanton and M. Allman, "Rfc 3708: Using tcp duplicate selective acknowledgement (dsacks) and stream control transmission protocol (sctp) duplicate transmission sequence numbers (tsns) to detect spurious retransmissions," *Request for Comments, IETF*, 2004. 29

[27] S. Ladha, S. Baucke, R. Ludwig, and P. Amer, "On making sctp robust to spurious retransmissions," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 2, pp. 123–135, 2004. 29

[28] K. Zheng, M. Liu, Z. Li, and G. Xu, "Shop: An integrated scheme for sctp handover optimization in multihomed environments," in *Proc. IEEE Global Communication Conf.*, 2008, pp. 1–5. 30

[29] D. Kim and S. Koh, "Performance enhancement of msctp for vertical handover across heterogeneous wireless networks," *Int. J. Commun. Syst.*, vol. 22, no. 12, pp. 1573–1591, 2009. 30

[30] J. Iyengar, P. Amer, and R. Stewart, "Concurrent multipath transfer using sctp multihoming over independent end-to-end paths," *IEEE/ACM Trans. Netw.*, vol. 14, no. 5, pp. 951–964, 2006. 31, 33, 35, 54, 59

[31] G. Ye, T. Saadawi, and M. Lee, "Ipcc-sctp: An enhancement to the standard sctp to support multi-homing efficiently," in *Proc. IEEE Intl. Conf. on Performance, Computing, and Communications*, 2004, pp. 523–530. 32, 59

[32] A. El Al, T. Saadawi, and L. M., "Ls-sctp: A bandwidth aggregation technique for stream control transmission protocol," *Comput. Commun.*, vol. 27, no. 10, pp. 1012–1024, 2004. 32, 59

[33] J. R. Iyengar, P. D. Amer, and R. Stewart, "Receive buffer blocking in concurrent multipath transfer," in *Proc. IEEE Global Communication Conf.*, vol. 1, 2005, pp. 121–126. 33

[34] ——, "Performance implications of a bounded receive buffer in concurrent multipath transfer," *Comput. Commun.*, vol. 30, no. 4, pp. 818–829, 2007. 33, 59, 103

[35] J. Liu, H. Zou, J. Dou, and Y. Gao, "Reducing receive buffer blocking in concurrent multipath transfer," in *Proc. IEEE Intl. Conf. on Circuits and Systems for Communications*, 2008, pp. 367–371. 34

[36] E. Yilmaz, N. Ekiz, P. Natarajan, P. Amer, J. Leighton, F. Baker, and R. Stewart, "Throughput analysis of non-renegable selective acknowledgments (nr-sacks) for sctp," *Comput. Commun.*, vol. 33, no. 16, pp. 1982–1991, 2010. 34

[37] P. Natarajan, N. Ekiz, P. Amer, and R. Stewart, "Concurrent multipath transfer during path failure," *Comput. Commun.*, vol. 32, no. 15, pp. 1577–1587, 2009. 35

[38] W. Yang, H. Li, and J. Wu, "Pam: Precise receive buffer assignment method in transport protocol for concurrent multipath transfer," in *Proc. Intl. Conf. on Communications and Mobile Computing*, 2010, pp. 413–417. 35

[39] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling tcp reno performance: a simple model and its empirical validation," *IEEE/ACM Trans. Netw.*, vol. 8, no. 2, pp. 133–145, 2000. 35, 81, 98

[40] W. Yang, H. Li, F. Li, Q. Wu, and J. Wu, "Rps: range-based path selection method for concurrent multipath transfer," in *Proc. IEEE Intl. Wireless Communications and Mobile Computing Conf.*, 2010, pp. 944–948. 35

[41] C. Casetti, W. Gaiotto, and D. di Elettronica, "Westwood sctp: Load balancing over multipaths using bandwidth-aware source scheduling," in *Proc. IEEE Vehicular Technology Conf.*, vol. 4, 2004, pp. 3025–3029. 36

[42] S. Mascolo, L. Grieco, R. Ferorelli, P. Camarda, and G. Piscitelli, "Performance evaluation of westwood+ tcp congestion control," *Perform. Evaluation*, vol. 55, no. 1-2, pp. 93–111, 2004. 36

[43] V. Bui, W. Zhu, A. Botta, and A. Pescapé, "A markovian approach to multipath data transfer in overlay networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. In Press, Corrected Proof, 2010. 37

[44] R. Prasad, C. Dovrolis, M. Murray, and K. Claffy, "Bandwidth estimation: metrics, measurement techniques, and tools," *IEEE Network*, vol. 17, no. 6, pp. 27–35, 2003. 37

[45] R. Fracchia, C. Casetti, C. Chiasserini, and M. Meo, "Wise: Best-path selection in wireless multihoming environments," *IEEE Trans. Mob. Comput.*, vol. 6, no. 10, pp. 1130–1141, 2007. 37, 38

[46] K. Pentikousis, "Tcp in wired-cum-wireless environments," *IEEE Commun. Surv.*, vol. 3, no. 4, pp. 2–14, 2000. 38

[47] B. Sardar and D. Saha, "A survey of tcp enhancements for last-hop wireless networks," *IEEE Commun. Surv. Tutorials*, vol. 8, no. 3, pp. 20–34, 2006. 38

[48] K. Leung and V. Li, "Transmission control protocol (tcp) in wireless networks: issues, approaches, and challenges," *IEEE Commun. Surv. Tutorials*, vol. 8, no. 4, pp. 64–79, 2006. 38

[49] S. Liu, S. Yang, and W. Sun, "Collaborative sctp: A collaborative approach to improve the performance of sctp over wired-cum-wireless networks," in *Proc. IEEE Intl. Conf. on Local Computer Networks*, 2004, pp. 276–283. 39

[50] S. Floyd, "Tcp and explicit congestion notification," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 24, no. 5, p. 23, 1994. 39

[51] G. Ye, T. Saadawi, and M. Lee, "Improving stream control transmission protocol performance over lossy links," *IEEE J. Sel. Areas Commun.*, vol. 22, no. 4, 2004. 40

[52] E. Ribeiro and V. Leung, "Minimum delay path selection in multi-homed systems with path asymmetry," *IEEE Commun. Lett.*, vol. 10, no. 3, pp. 135–137, 2006. 41

[53] A. Argyriou and V. Madisetti, "Bandwidth aggregation with sctp," in *Proc. IEEE Global Communication Conf.*, vol. 7, 2003. 41

[54] W. Xing, H. Karl, A. Wolisz, and H. Müller, "M-sctp: Design and prototypical implementation of an end-to-end mobility concept," in *Proc. Intl. Workshop on The Internet Challenge: Technology and Applications*, 2002. 43

[55] A. Kelly, G. Muntean, P. Perry, and J. Murphy, "Delay-centric handover in sctp over wlan," *Autom. Control Comput. Sci.*, vol. 49, no. 63, pp. 1–6, 2004. 43

[56] L. Bokor, Á. Huszák, and G. Jeney, "On sctp multihoming performance in native ipv6 umts-wlan environments," in *Proc. ICST Intl. Conf. Testbeds and Research Infrastructures for the Development of Networks and Communities*, 2009, pp. 1–10. 43

[57] M. Chiu and M. Bassiouni, "Predictive schemes for handoff privatization in cellular networks based on mobile positioning," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 3, pp. 510–522, 2000. 44

[58] D. Lee and Y. Hsueh, "Bandwidth-reservation scheme based on road information for next-generation cellular networks," *IEEE Trans. Veh. Technol.*, vol. 53, no. 1, pp. 243–252, 2004. 44

[59] N. Samaan and A. Karmouch, "A mobility prediction architecture based on contextual knowledge and spatial conceptual maps," *IEEE Trans. Mob. Comput.*, vol. 4, no. 6, pp. 537–551, 2005. 44

[60] W. Soh and H. Kim, "A predictive bandwidth reservation scheme using mobile positioning and road topology information," *IEEE/ACM Trans. Netw.*, vol. 14, no. 5, p. 1091, 2006. 44

[61] J. Olsén, "Stochastic modeling and simulation of the tcp protocol," Ph.D. dissertation, Department of Mathematics, Uppsala University, 2003. 81

[62] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the tcp congestion avoidance algorithm," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 27, no. 3, pp. 67–82, 1997. 81, 86

[63] R. Dunaytsev, Y. Koucheryavy, and J. Harju, "The pftk-model revised," *Comput. Commun.*, vol. 29, pp. 13–14, 2006. 81

[64] Z. Chen, T. Bu, M. Ammar, and D. Towsley, "Comments on modeling tcp reno performance: a simple model and its empirical validation," *IEEE/ACM Trans. Netw.*, vol. 14, no. 2, pp. 451–453, 2006. 81

[65] J. Hwang and C. Yoo, "Formula-based tcp throughput prediction with available bandwidth," *IEEE Commun. Lett.*, vol. 14, no. 4, pp. 363–365, 2010. 81

[66] A. Kumar, "Comparative performance analysis of versions of tcp in a local network with a lossy link," *IEEE/ACM Trans. Netw.*, vol. 6, no. 4, pp. 485–498, 1998. 81

[67] V. Firoiu and M. Borden, "A study of active queue management for congestion control," in *Proc. IEEE Intl. Conf. on Computer Communications*, vol. 3, 2000, pp. 1435–1444. 81

[68] C. Barakat, "Tcp/ip modeling and validation," *IEEE Network*, vol. 15, no. 3, pp. 38–47, 2001. 81

[69] N. Parvez, A. Mahanti, and C. Williamson, "An analytic throughput model for tcp newreno," *IEEE/ACM Trans. Netw.*, vol. 18, no. 2, pp. 448–461, 2010. 81

[70] C. Casetti and M. Meo, "An analytical framework for the performance evaluation of tcp reno connections," *Comput. Netw.*, vol. 37, no. 5, pp. 669–682, 2001. 81

[71] A. Wierman, T. Osogami, and J. Olsén, "A unified framework for modeling tcp-vegas, tcp-sack, and tcp-reno," in *Proc. IEEE Intl. Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, 2003, pp. 269–278. 81

[72] M. Garetto, R. Cigno, M. Meo, and M. Marsan, "Closed queueing network models of interacting long-lived tcp flows," *IEEE/ACM Trans. Netw.*, vol. 12, no. 2, pp. 300–311, 2004. 81

[73] S. Fu and M. Atiquzzaman, "Performance modeling of sctp multihoming," in *Proc. IEEE Global Communication Conf.*, vol. 2, 2005, pp. 786–791. 82

[74] T. Wallace and A. Shami, "An analytic model for the stream control transmission protocol," in *Proc. IEEE Global Communication Conf.*, 2010, pp. 1–5. 82

[75] S. Fortin-Parisi and B. Sericola, "A markov model of tcp throughput, goodput and slow start," *Perform. Evaluation*, vol. 58, no. 2, pp. 89–108, 2004. 82

[76] J. Padhye, V. Firoiu, and D. Towsley, "A stochastic model of tcp reno congestion avoidance and control," Department of Computer Science, University of Massachusetts, Tech. Rep. UMASS-CS-TR-1999-02, 1999. 82

[77] J. Bolot, "End-to-end packet delay and loss behavior in the internet," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 23, no. 4, pp. 289–298, 1993. 103

[78] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and S. Diot, "Packet-level traffic measurements from the sprint ip backbone," *IEEE Network*, vol. 17, no. 6, pp. 6–16, 2003. 103

[79] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley, "Measurement and classification of out-of-sequence packets in a tier-1 ip backbone," *IEEE/ACM Trans. Netw.*, vol. 15, no. 1, pp. 54–66, 2007. 103

[80] G. Strang, *Linear algebra and its applications.* Hartcourt Brace Jovanovich College Publishers, 1988. 106

[81] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock, "Host-to-host congestion control for tcp," *IEEE Commun. Surv. Tutorials*, vol. 12, no. 3, pp. 304–342, 2010. 110

[82] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control (bic) for fast long-distance networks," in *Proc. IEEE Intl. Conf. on Computer Communications*, vol. 4, 2004, pp. 2514–2524. 110

[83] D. Kliazovich, F. Granelli, and D. Miorandi, "Logarithmic window increase for tcp westwood+ for improvement in high speed, long distance networks," *Comput. Netw.*, vol. 52, no. 12, pp. 2395–2410, 2008. 110

[84] S. Ha, I. Rhee, and L. Xu, "Cubic: A new tcp-friendly high-speed tcp variant," *ACM SIGOPS Operating Syst. Rev.*, vol. 42, no. 5, pp. 64–74, 2008. 110

[85] R. Wang, K. Yamada, M. Y. Sanadidi, and M. Gerla, "Tcp with sender-side intelligence to handle dynamic, large, leaky pipes," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 2, pp. 235–248, 2005. 110

[86] B. Allcock, J. Bester, J. Bresnahan, A. Chervenak, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, S. Tuecke, and I. Foster, "Secure, efficient data transport and replica management for high-performance data-intensive computing," in *Proc. IEEE Symposium on Mass Storage Systems and Technologies*, 2001, pp. 13–28. 111

[87] J. Semke, J. Mahdavi, and M. Mathis, "Automatic tcp buffer tuning," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 28, no. 4, pp. 315–323, 1998. 111

[88] E. Weigle and W. Feng, "Dynamic right-sizing: A simulation study," in *Proc. IEEE Intl. Conf. on Computer Communications and Networking*, 2001, pp. 152–158. 111

[89] T. Dunigan, M. Mathis, and B. Tierney, "A tcp tuning daemon," in *Proc. ACM/IEEE Conf. on Supercomputing*, 2002, pp. 1–16. 111

[90] R. Prasad, M. Jain, and C. Dovrolis, "Socket buffer auto-sizing for high-performance data transfers," *J. Grid Comput.*, vol. 1, no. 4, pp. 361–376, 2003. 111, 114

[91] G. Lawton, "4g: Engineering versus marketing," *Computer*, vol. 44, no. 3, pp. 14–16, 2011. 111

# Appendix A
# The On-demand Scheduler

The search algorithm for ODS is now presented in Fig. A.1. For purposes of explanation, the algorithm is broken into four parts, namely: (1) Search Initialization, (2) Simulating Acknowledgements, (3) Calculating ETAs, and (4) Simulating Transmissions.

The purpose of part (1) is to simply introduce and initialize key variables necessary for the search. The effective algorithm, however, really begins in part (2). First, line 1 checks if all packets in the send buffer (SBUF) have been transmitted, or if none are outstanding; if either is true the search will exit, and no packet will be transmitted to the destination. Otherwise, line 4 assumes a packet will be acknowledged at a time equal to the next earliest ETA. Initially, the time of the next transmission opportunity is set to that of the earliest ETA. Since it is possible a packet could be late, i.e. if $j.eta < t^{\mathrm{NS}}$, the acknowledgement is assumed to be imminent; so $t = t^{\mathrm{NS}}$. The remaining lines simply follow normal acknowledgement procedures; increasing a destination's *partial.bytes.acked* (PBA), checking if a congestion window can be increased, and removing cumulative packets from the SBUF.

Lines 2-5 of part (3) compute new ETAs for the next cumulative packet $i$ at time $t$. If $d^{\mathrm{NS}}$ offers the soonest ETA, then a packet is found and the current search is finished. If other destinations can still receive packets before $d^{\mathrm{NS}}$, then the algorithm removes any destinations in $D'$ that cannot receive $i$ before $d^{\mathrm{NS}}$ and waits for another acknowledgement.

(1) Search Initialization:

let $d^{\mathrm{NS}}$ be the destination looking for a packet at the start of a new search;
let $t^{\mathrm{NS}}$ be the starting time of a new search;
let $Q$ be a copy of the SBUF at the start of a new search;
let $D$ be a copy of all destination state at the start of a new search;
let $D'$ be a the set of destinations with earlier ETAs at the start of a new search;
goto (2);

(2) Simulating Acknowledgements:

**if** all packets in $Q$ have been transmitted or acknowledged **then**
    exit search with failure;
**else**
    let $j$ be the packet in $Q$ with the earliest ETA such that $j.sent$ = TRUE and $j.acked$ = FALSE;
    set $t = j.eta$;
    **if** $t < t^{\mathrm{NS}}$ **then**
        set $t = t^{\mathrm{NS}}$;
    **end if**
    set $d = j.dest$;
    increment $d.pba$;
    update $d.cwnd$;
    remove all cumulative packets from $Q$;
**end if**
goto (3);

(3) Calculating ETAs:

let $i$ be the packet in $Q$ with the lowest sequence number such that $i.sent$ = FALSE;
compute $A(i, d^{\mathrm{NS}}, t^{\mathrm{NS}})$;
**for** each destination $d$ in $D'$ **do**
    compute $A(i, d, t)$;
**end for**
**if** $d^{\mathrm{NS}}$ has the lowest ETA at time $t$ **then**
    return with $s$
**else**
    remove any destinations from $D'$ with later ETAs than $d^{\mathrm{NS}}$;
**end if**
goto (4);

(4) Simulating Transmissions:

let $D_t^{\mathrm{Tx}}$ be the set of destinations at time $t$ with a transmission opportunity;
**if** $D_t^{\mathrm{Tx}} \neq \{\emptyset\}$ **then**
    **if** $d_{\min} \neq d_{\min}^{\mathrm{Tx}}$ **then**
        start a new search for $d_{\min}^{\mathrm{Tx}}$ using $Q$, $D$, and $D'$ at time $t$;
        goto (1);
    **end if**
    set $i.dest = d_{\min}^{\mathrm{Tx}}$;
    set $i.eta = A(i, d_{\min}^{\mathrm{Tx}}, t)$;
    set $i.sent$ = TRUE;
**end if**
goto (2);

Figure A.1: The on-demand scheduler (ODS).

Before simulating another acknowledgement, however, line 2 of part (4) checks if any other destination can transmit at time $t$. Assuming $D_t^{\mathrm{Tx}}$ is a subset of $D'$, such that all destinations in $D_t^{\mathrm{Tx}}$ have a transmission opportunity at $t$; then if $D_t^{\mathrm{Tx}}$ is not empty, at least one destination can still receive packets ahead of $d^{\mathrm{NS}}$. Line 3 selects the best destination first; if $d_{\min} \neq d_{\min}^{\mathrm{Tx}}$, then a new search is started for $d_{\min}^{\mathrm{Tx}}$ (lines 3-6). On the contrary, if the destination with the earliest ETA also has a transmission opportunity, the algorithm remains in the current search; simulating a transmission, then returning to (2).

# Appendix B

# Congestion Window Update Policy

```
1: let d be the destination updating its CWND;
2: let d.BDP be the bandwidth delay product of destination d;
3: let d.CWND be the congestion window of destination d;
4: let d.PBA be the value of partial.bytes.acked for destination d;
5: let CWND_sum be the sum of all congestion windows;
6: let RWND_max be the maximum size of the receive window;
7: let D be the set of all destinations with RTTs higher than d in descending order;
8: if d.PBA ≥ d.CWND then
9:    if CWND_sum < RWND_max then
10:      if d.CWND < d.BDP then
11:         increment d.CWND;
12:      else if d.CWND > d.BDP then
13:         decrement d.CWND;
14:      end if
15:   else if d.CWND < d.BDP and D ≠ {∅} then
16:      for all i in D do
17:         if i.CWND > 1 then
18:            decrement i.CWND;
19:            increment d.CWND;
20:            break;
21:         end if
22:      end for
23:   end if
24: end if
```

Figure B.1: $policy_2$: a new congestion window update policy for CMT.

# Curriculum Vitae

**Name:** Thomas Daniel Wallace

**Education:**

B.Eng. 2004
Lakehead University
Thunder Bay, Ontario, Canada

M.E.Sc. 2007
The University of Western Ontario
London, Ontario, Canada

Ph.D., Engineering Sc. 2012
The University of Western Ontario
London, Ontario, Canada

**Publications**

Refereed Journal Papers:

[1] T. D. Wallace, A. Shami, and C. Assi, "Advance lightpath reservation for WDM networks with dynamic traffic," *Journal of Optical Networking*, vol. 6, no. 7, pp. 913-924, 2007.

[2] T. D. Wallace, A. Shami, and C. Assi, "Scheduling advance reservation requests for wavelength division multiplexed networks with static traffic demands," *IET Communications*, vol. 2, no. 8, pp. 1023-1033, 2008.

[3] T. D. Wallace, and A. Shami, "A review of multihoming issues using the stream control transmission protocol," *IEEE Communications Surveys & Tutorials*, no. 99, pp. 1-14, 2011.

Refereed Conference Proceedings:

[1] T. D. Wallace, and A. Shami, "Advanced lightpath reservation in WDM networks," *IEEE International Conference on Computer Communications International Conference* (INFOCOM), pp. 1-3, 2006.

[2] T. D. Wallace, and A. Shami, "Connection management algorithm for advance lightpath reservation in WDM networks," *International Conference on Broadband Communications, Networks, and Systems* (BROADNETS), pp. 837-844, 2007.

[3] T. D. Wallace, and A. Shami, "An analytic model for the stream control transmission protocol," *IEEE Global Telecommunications Conference* (GLOBECOM), pp. 1-5, 2010.

[4] T. D. Wallace, and A. Shami, "On-demand scheduling for concurrent multipath transfer under delay-based disparity," *International Wireless Communications and Mobile Computing Conference* (IWCMC), 2012.

Masters Thesis:

[1] T. D. Wallace, "Advance lightpath reservation for wavelength division multiplexed networks," *The University of Western Ontario*, 2006, supervised by Dr. A. Shami.

**Awards and Scholarships**

[1] Ontario Graduate Scholarship (OGS), 2009, 2010.

[2] Ontario Graduate Scholarship in Science and Technology (OGSST), 2008.

[3] Graduate Thesis Research Award, The University of Western Ontario, 2007, 2008.

[4] Student Travel Grant, The University of Western Ontario, 2006, 2007, 2010.

[5] Award for Outstanding Presentation at the ECE Graduate Symposium, The University of Western Ontario, 2009.

[6] Western Engineering Graduate Entrance Scholarship, The University of Western Ontario, 2007.

[7] Western Engineering Scholarship, The University of Western Ontario, 2005-2010.

[8] First Class Standing, Lakehead University, 2004.

[9] Entrance Scholarship, Lakehead University, 2000.